



AFRL-RY-WP-TR-2016-0030

SPARSE DISTRIBUTED REPRESENTATION & HIERARCHY: KEYS TO SCALABLE MACHINE INTELLIGENCE

Gerard (Rod) Rinkus, Greg Leshner, Jasmin Leveille, and Oliver Layton

Neurithmic Systems, LLC

**APRIL 2016
Final Report**

Approved for public release; distribution unlimited.

See additional restrictions described on inside pages

© 2016 Neurithmic Systems, LLC

STINFO COPY

**AIR FORCE RESEARCH LABORATORY
SENSORS DIRECTORATE
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-7320
AIR FORCE MATERIEL COMMAND
UNITED STATES AIR FORCE**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nationals.

AFRL-RY-WP-TR-2016-0030 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

// Signature//

KERRY L. HILL
Program Manager
Advanced Sensor Components Branch
Aerospace Components & Subsystems Division

// Signature//

BRADLEY J. PAUL, Chief
Advanced Sensor Components Branch
Aerospace Components & Subsystems Division

// Signature//

MARK G. SCHMITT
Chief (Acting)
Aerospace Components & Subsystems Division
Sensors Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

*Disseminated copies will show “//Signature//” stamped or typed above the signature blocks.

REPORT DOCUMENTATION PAGE					Form Approved OMB No. 0704-0188	
<p>The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.</p>						
1. REPORT DATE (DD-MM-YY) April 2016		2. REPORT TYPE Final		3. DATES COVERED (From - To) 29 April 2013 – 30 November 2015		
4. TITLE AND SUBTITLE SPARSE DISTRIBUTED REPRESENTATION & HIERARCHY: KEYS TO SCALABLE MACHINE INTELLIGENCE				5a. CONTRACT NUMBER FA8650-13-C-7342		
				5b. GRANT NUMBER		
				5c. PROGRAM ELEMENT NUMBER 61101E		
6. AUTHOR(S) Gerard (Rod) Rinkus, Greg Lesher, Jasmin Leveille, and Oliver Layton				5d. PROJECT NUMBER 1000		
				5e. TASK NUMBER N/A		
				5f. WORK UNIT NUMBER Y0Z9		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Neurithmic Systems, LLC 468 Waltham Street Newton, MA 02465				8. PERFORMING ORGANIZATION REPORT NUMBER		
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory Sensors Directorate Wright-Patterson Air Force Base, OH 45433-7320 Air Force Materiel Command United States Air Force				10. SPONSORING/MONITORING AGENCY ACRONYM(S) AFRL/Rydi		
				11. SPONSORING/MONITORING AGENCY REPORT NUMBER(S) AFRL-RY-WP-TR-2016-0030		
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited.						
13. SUPPLEMENTARY NOTES DISTAR Case Number: 25997, cleared 9 March 2016. © 2016 Neurithmic Systems, LLC. This work was funded in whole or in part by Department of the Air Force Contract FA8650-13-C-7342. The U.S. Government has for itself and others acting on its behalf a paid-up, nonexclusive, irrevocable worldwide license to use, modify, reproduce, release, perform, display, or disclose the work by or on behalf of the U. S. Government. Report contains color.						
14. ABSTRACT We developed and tested a cortically-inspired model of spatiotemporal pattern learning and recognition called Sparsey. Sparsey is a hierarchical model allowing an arbitrary number of levels consisting of coding modules that code information, specifically particular spatiotemporal input moments using sparse distributed representations (SDRs). The modules are called “macs” as they are proposed as analogs of the canonical cortical processing module known as macrocolumns. Sparsey differs from mainstream neural models, e.g., Deep Learning, in many ways including: a) it uses single-trial, Hebbian learning rather than incremental, many-trial, gradient-based learning; and b) it multiplicatively combines bottom-up, top-down, and horizontal, evidence at every unit (neuron) in every mac at every level on every time step during learning and inference (retrieval). However, Sparsey’s greatest distinguishing characteristic is that it does both learning (storage) and retrieval of the best matching stored input in time that remains constant regardless of how many patterns (how much information) has been stored. Thus, it has excellent scaling potential to “Big Data”-sized problems. We conducted numerous studies establishing basic properties and capacities, culminating in demonstration of 67% classification accuracy on the Weizmann data set, accomplished with 3.5 minutes training time, with no machine parallelism and almost no software optimization.						
15. SUBJECT TERMS video event recognition, neural network, sparse distributed representations, machine intelligence, probabilistic models, hierarchy, cortical model, brain model						
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT: SAR	18. NUMBER OF PAGES 232	19a. NAME OF RESPONSIBLE PERSON (Monitor) Kerry Hill	
a. REPORT Unclassified	b. ABSTRACT Unclassified	c. THIS PAGE Unclassified			19b. TELEPHONE NUMBER (Include Area Code) N/A	

Table of Contents

Section	Page
LIST OF FIGURES	IV
LIST OF TABLES	VIII
1. ACKNOWLEDGEMENT.....	1
2. SUMMARY	2
3. INTRODUCTION.....	4
3.1 The First Key: Sparse Distributed Representations	6
3.2 The Second Key: Hierarchy	7
3.3 Putting the Two Keys Together.....	8
3.4 Final State of the Research	11
4. METHODS, ASSUMPTIONS, AND PROCEDURES	14
4.1 Overview of Model Architecture	14
4.2 The Algorithm	15
4.2.1. CSA: Learning Mode	16
4.2.1.a. Step 1: Determine if the Mac will become Active.....	17
4.2.1.b. Step 2: Compute Raw U, H, and D-Summations for each Cell, i, in the Mac..	18
4.2.1.c. Step 3: Normalize and Filter the Raw Summations.....	18
4.2.1.d. Step 4: Compute Overall Local Support for Each Cell in the Mac	20
4.2.1.e. Step 5: Compute the Number of Competing Hypotheses that will be Active in the Mac once the Final Code for this Frame is Activated	21
4.2.1.f. Step 6: Compute Correction Factor for MCHs to be Applied to Efferent Signals from this Mac	23
4.2.1.g. Step 7: Determine the Maximum Local Support in each of the Mac's CMs....	24
4.2.1.h. Step 8: Compute the Familiarity of the Mac's Overall Input	24
4.2.1.i. Step 9: Determine the Expansivity/Compressivity of the I/O Function to be used for the Second and Final Round of Competition within the Mac's CMs	25
4.2.1.j. Step 10: Apply the Modulated Activation Function to all the Mac's Cells, Resulting in a Relative Probability Distribution of Winning over the Cells of each CM...	28
4.2.1.k. Step 11: Convert Relative Win Probability Distributions to Absolute Distributions	29
4.2.1.l. Step 12: Pick Winners in the Mac's CMs, i.e., Activate the SDC.....	29
4.2.1.m. Learning Policy and Mechanics	31
4.2.2. CSA: Retrieval Mode	33
4.2.3. CSA: Simple Retrieval Mode.....	40
5. RESULTS.....	42
5.1 Individual Macs Implement SISC	42
5.2 Simple Features Support High Class Accuracy	43
5.3 Sanity Tests (Test Set = Train Set).....	43
5.3.1. Sanity and Noisy Recognition Tests with Edge-Filtered Videos	44
5.3.2. Lower Resolution Weizmann Edge.....	46
5.3.3. Sanity Test: 3-Level Model Revealing DCCI Principle.....	47
5.4 Family-Resemblance Classification Style.....	52
5.5 Experiments with More Powerful Input Feature, HOFs	55
5.6 Principles/Mechanism of Invariance	60

Section	Page
5.7 Application to Purely Spatial Pattern Recognition Problems	63
5.7.1. Re-use of Existing Knowledge in Hierarchical Networks	75
5.8 Large-Scale Episodic Memory Capacity Study	85
5.9 Effect of π -bounds \times U-RF Interaction on Capacity/Accuracy.....	86
5.9.1. Effect of U-RF Sizes/Overlaps, and π -bounds on Capacity/Accuracy	92
6. DISCUSSION	98
6.1 Regarding Sparsey's Information Storage Capacity	98
6.2 Supervised Learning via Cross-Modal Unsupervised Learning.....	99
6.3 Hierarchical Exemplar-Based Categorization	104
6.4 Optimal Normalization Thresholds	117
6.5 Fraction of Represented Features Should Remain near 100% at All Levels	118
6.6 Hierarchical Compression	122
6.7 Trace Accuracy can be Quite Low, While Supporting High Classification Accuracy.....	123
6.8 Minimize the Number of Post-Quiescent Mac Activations	126
6.9 Correct Cells are Correlated, Incorrect Cells are Not	127
7. CONCLUSIONS	130
7.1 Importance of Unitary Explanation of Episodic and Semantic Memory	131
8. RECOMMENDATIONS	132
8.1 Applying Supervision at Multiple Scales	134
8.2 Outstanding Questions Regarding Parameter Settings.....	136
8.2.1. Backoff	136
9. REFERENCES	138
APPENDIX A - EDGE FILTERING DETAIL	142
A.1 The Video Datasets	142
A.2 Preprocessing Protocol	143
1. Stage 1: Gaussian Smoothing.....	144
2. Stage 2: Gabor Edge Detection	145
3. Stage 3: Orientation and Phase Superposition	146
4. Stage 4: Surround Suppression.....	147
5. Stage 5: Edge Thinning	149
6. Stage 6: Hysteresis Thresholding	149
7. Stage 7: Suppression Slope	150
8. Future Considerations.....	151
APPENDIX B - SISC PROPERTY	153
B.1 Study 1: Basic SISC Properties	153
B.2 Explanation of the Code Selection Algorithm (CSA)	158
B.3 Study 2: SISC Properties as Function of V-to- μ Mapping Parameter, π	167
B.4 Study 3: Arbitrary Temporal Context Window Length of SISC Property.....	170
B.5 Conclusions	172
APPENDIX C - ROBUST CAPABILITY TO HANDLE COMPLEX SEQUENCES	174
APPENDIX D - MECHANISMS UNDERLYING INVARIANT RECOGNITION	183
APPENDIX E - RELATION OF INPUT ACCURACY MEASURE TO RECOGNITION ACCURACY	194
E.1 Example Computation for a Level 2 Mac	194

Section	Page
E.2 Sample Input Accuracy from a Simulation	195
E.3 Distribution of RA% in a Feedforward Network	195
APPENDIX F - SPATIOTEMPORAL COMPOSITIONAL HIERARCHIES IN SPARSEY	197
APPENDIX G - STATEMENT OF WORK.....	209
G.1 Task List	209
10. SYMBOL TABLE	218
LIST OF ABBREVIATIONS AND ACRONYMS	220

LIST OF FIGURES

Figure	Page
Figure 1: Comparison of Localist and SDC-based Hierarchical Vision Models.....	9
Figure 2: Notional Mapping of Sparsey to Brain.....	11
Figure 3: Snapshot of Best Performing Model in Study.....	13
Figure 4: The Currently Active Mac Code Simultaneously Physically Functions as the Entire Likelihood Distribution over all Hypotheses Stored in the Mac.....	15
Figure 5: Generic “Circuit Model” for Reference in Describing Some Steps of the CSA.....	17
Figure 6: Mac Normalization Policy Handles Inputs with Varying Numbers of Features.....	20
Figure 7: Illustration of Method for Handling MCHs	22
Figure 8: Consequences on Probabilistic Code Selection of Amount of Prior Learning and Varying Characteristics of <i>G</i> -based Sigmoid Transform.....	27
Figure 9: More Rightward Sigmoid Inflection Point Protects Against Mounting Crosstalk.....	28
Figure 10: The “Weight Table” Indexed by Age and Permanence	32
Figure 11: Formation of Hierarchical Spatiotemporal Memory Trace, Unrolled in Time	35
Figure 12: Motivation for the Back-Off Strategy for Computing <i>G</i> in Retrieval Mode.....	37
Figure 13: Back-Off Allows Internal State to Remain Synchronized with Nonlinearly Time-Warped Instances of Known Snippets	39
Figure 14: Demonstration of SISC Property for 4-item-long Sequences	42
Figure 15: Original and Preprocessed Frames of a Weizmann Snippet	43
Figure 16: Original Edge-filtered Frame and Corresponding 40% Noisy Frame.....	44
Figure 17: Snapshot of 2-level Model Used in Sanity Test Study.....	45
Figure 18: Lower Spatial Resolution (42x60) Original and 20% Noisy Frame	46
Figure 19: Model Achieving ~100% Class Accuracy in Weizmann Sanity Test.....	49
Figure 20: Example Showing Richness of Sparsey’s Modular, i.e., Mac-Based, Representation of the Hierarchical Part-Whole Structure of Inputs	51
Figure 21: Substantially Different Though Overlapped Sets of Active L1 and L2 Macs Between Original and Noisy Version of a Snippet.....	53
Figure 22: Model Achieving ~100% Accuracy on HOF-filtered Weizmann Dataset.....	56
Figure 23: 4-Frame HOF Snippet and Corresponding Spatiotemporal Memory Trace	57
Figure 24: Original and 40% Noisy Frames of the Ido_Bend Snippet.....	59
Figure 25: Demonstration of Invariant Recognition.....	61
Figure 26: Demonstration of Robust Invariance.....	63
Figure 27: Sample of Original MNIST Data and Our Preprocessed Versions	64
Figure 28: The 10 Instances of Digit ‘5’ Used in this Experiment.....	64
Figure 29: The 4-level Model with Preprocessed MNIST Digit ‘8’ Active in Input Level	65
Figure 30: The 10 Instances of All 10 Digits with Superimposed V0 Aperture Grid	68
Figure 31: A Second Model Used in MNIST Digit Experiments.....	69
Figure 32: Chart Representation of Table 9.....	70
Figure 33: Learned Bases in Four V1 Macs for MNIST Experiment.....	71
Figure 34: Approximate Bases for All Six V2 Macs	75
Figure 35: Sketch of Model Used in Study 1.....	76
Figure 36: Ten Digit ‘4’ Instances in Train/Test Set and Detailed SDR Codes for All Active Macs for Instance 2 Learning Trial	77

Figure	Page
Figure 37: The Two Prior Learning Trials with Identical Inputs in Some Apertures and Codes in the Associated L1 Macs	79
Figure 38: Detailed Pixel Patterns Present for Two Instances and Compositional Views of Featural Hierarchies Comprising Overall Concepts	81
Figure 39: Full Hierarchical Codes (Memory Traces) for Instance 0 of Digit ‘2’ (top) and Instance 0 of Digit ‘4’ (bottom)	82
Figure 40: A Study of Invariances Across Three Scales	84
Figure 41: The 8-level Network Used in Large-Scale Episodic Memory Study	86
Figure 42: Three of the 8-Frame 32x32 Snippets Used in Section 5.9 Studies	87
Figure 43: Snapshot of 7-Level Model Used in Section 5.9 Studies	89
Figure 44: Gross Architectures of Models Used in Studies of Table 13-Table 15	94
Figure 45: Mapping from High-Dimensional Visual Input Space to Notional V1 Mac	100
Figure 46: Implementation of Supervised Learning as an Instance of Cross-Modal Unsupervised Association	101
Figure 47: Abstraction-Based vs. Exemplar-Based Categorization Schemes	103
Figure 48: A Multi-level Sparsey Network Implements a Hierarchical, Exemplar-Based Clustering/Categorization Scheme	105
Figure 49: Network of Figure 48 but Emphasizing One Particular Active L2 Mac	107
Figure 50: Network of Prior Figures but Emphasizing the L3 Mac	108
Figure 51: All Frames of the Input Set of Study 1 of Technical Report 7	109
Figure 52: 3D View Emphasizing the Hierarchical, Compositional Nature of the Multi-Level Representations	110
Figure 53: 2D View of Features Comprising the Particular Top-Level Feature Present in this Ongoing Example	111
Figure 54: Correspondence of 3D and 2D Visualizations of Compositional Hierarchies	112
Figure 55: Hierarchical Compositional Memory Traces of First Four Moments of Figure 51	113
Figure 56: Alternate View of Figure 55c in the Style of Figure 53	114
Figure 57: Three Top-Level Feature Instances and Hypothetical Inputs with Varying Pixel-Wise and Semantic Relatedness to Those Instances	115
Figure 58: Spatiotemporal Generalization of Figure 48 Showing that SDR Codes Actually Represent Spatiotemporal Features	117
Figure 59: A Case in which a Higher U Normalization Threshold Would Yield Substantially Higher Correctness of the Reinstated Code	118
Figure 60: Surviving Input Features at All Internal Levels on Four Example Frames	120
Figure 61: Illustration of Overlapping L1 U-RFs	121
Figure 62: Illustration of Progressive Code Compression up Through the Model’s Levels	123
Figure 63: Example of Highly accurate Recognition at Higher Level Despite Highly Inaccurate Recognition at Lower Level	125
Figure 64: Demonstration of the Large Fraction of All L1 Activations that are PQAs	127
Figure 65: Example of Partial Recovery of Code Accuracy Despite Completely Incorrect Inputs Due to Second Order Correlations Resulting from SISC Property	129
Figure 66: Illustration of Applying Supervised learning at Multiple Scales in the Network of Figure 35	135

Figure	Page
Figure 67: Example Gabor Wavelets Used in Our Preprocessing	145
Figure 68: Gradient Image Frames Computed by Gabor Wavelet Filtering	146
Figure 69: Gabor Wavelet Image Gradients with Different Parameter Values for the Spatial Wavelength, Aspect Ratio, and Bandwidth	147
Figure 70: Illustration of Texture Suppression Kernel w_σ	148
Figure 71: Effects of Texture Surround Suppression on Image gradient of Sample Frame	148
Figure 72: The Results of Edge Thinning by Non-Maxima Suppression	149
Figure 73: Binarization Results Following Hysteresis Thresholding	150
Figure 74: Final Binarized Stage 7 Output Following Application of Suppression Slope	151
Figure 75: Comparison of Binarization Without (top) and With (bottom) Slope Suppression .	151
Figure 76: Six 2-Frame Sequences Used in Study 1 and Codes Assigned to S0	154
Figure 77: Portrayal of the Spatiotemporal SISC Property	156
Figure 78: Graphical Explanation of How Winners are Chosen in the CMs and Thus, How Entire Codes are Chosen	161
Figure 79: Graphical Explanation of Code Selection on the Second Moment, [AX], of the Second Presentation of S0	164
Figure 80: Graphical Explanation of Code Selection on First Moment, [E], of S3=[EX]	166
Figure 81: Graphical Explanation of Code Selection 2 nd Moment, [EX], of S3=[EX]	167
Figure 82: Presentation of Moment [EX] when $\pi = 100$	169
Figure 83: (a) Library of 16 Input Items from which Train and Test Sequences are Constructed and (b) The Input Set for Study 3	170
Figure 84: Experimental Protocol For Demonstrating that Spatiotemporal Similarity (G) Reflects the Whole History Leading up to the Current Moment	171
Figure 85: Demonstration of Spatiotemporal SISC Property for 4-Item-Long Sequences	172
Figure 86: The Four Sequences Comprising the Train/Test Set for the Complex Sequence Study and the Model Used in the Study	174
Figure 87: The Situation in the Network when the First Frame of Seq. 1 is Presented	175
Figure 88: The Situation in the Network on Learning Moment (1,1)	177
Figure 89: Comparison of Two Recognition Moments, (1,0) and (4,7), in which the Input Pattern is the Horizontal Bar Across the Bottom Row of L0	179
Figure 90: All Three Relevant Codes Activated on Recognition Moment (1,0) and Detailed Trace Information for the L2 Mac Showing Presence of Multiple Competing Hypotheses	180
Figure 91: Correct Resolution of Multiple Competing Hypotheses based on Temporal Context Signals Mediated by H and D Inputs	181
Figure 92: The Overall Recognition Performance for All Four Sequences	182
Figure 93: Demonstration of Invariant Recognition	184
Figure 94: Invariance: The Same Code Activates in M_{402}^2 Despite Very Different Immediate Input Patterns from L1	186
Figure 95: Examples of Invariances Learned	187
Figure 96: Invariant Recognition at L4 Mac M_{152}^4	189
Figure 97: Invariant Recognition at L6 Mac M_{19}^6	191
Figure 98: The Portions of the Input, i.e., <i>Features</i> or <i>Parts</i> , Coded by Each of the 16 Active L6 Macs on Frame 7 of the Input Snippet	193
Figure 99: Distribution of RA% as a Function of IA in Level 2 Macs	195

Figure	Page
Figure 100: Sets of Active Macs Cross Levels and Frames While Processing an 8-Frame Snippet of an “Extend Arm” Event, Showing Dropping Out of Features At Higher Levels	198
Figure 101: Illustration of Redundant Representation of Features (Parts)	199
Figure 102: 3D Version of Figure 101’s Center Panel to Clarify the Exposition.....	200
Figure 103: L2 Features are Larger and More Complex than L1 Features	201
Figure 104: Detail of the Featural Transform Between L2 and L3	202
Figure 105: L4 Representation of Frame 4 Consisting of four Large features, Some of which are Fairly Complex.....	204
Figure 106: Progressively Larger-Scale, More Complex, and More Overlapped Features at Successive Network Levels (Starting with L2)	206
Figure 107: A Visualization Clarifying the U-RF structure Across Network Levels	208
Figure 108: Prospective 2-Modality Fusion Architecture Combining Edge & HOF Features	217

LIST OF TABLES

Table	Page
Table 1. Classification Results on Weizmann Data as Function of SVM Cost Parameter.....	12
Table 2. The CSA during Learning.....	30
Table 3. Performance of 2-level Model on Weizmann Edge Snippets.....	45
Table 4. Model Performance on Low Space-Time Resolution Weizmann Edge Snippets	47
Table 5: 3-Level Model Performance in “Train = Test” Condition	48
Table 6. 3-Level Model Performance in “Test = 20% Noisy Train” Condition.....	54
Table 7. Performance of 2-level Model on “Test = Train” Condition.....	58
Table 8. Recognition Accuracy Results for Experiment 1	66
Table 9. D-summations at all Category Units for Test Trials for the 10 Instances of ‘0’	70
Table 10. Results of 64x64 Snippet Episodic Recognition Memory Study	85
Table 11. Recognition Accuracy Results for this Study	88
Table 12. Recognition Accuracy Results for Section 5.9 Studies	91
Table 13. Recognition Accuracy Results for this Study	92
Table 14. Recognition Accuracy Results for this Study	95
Table 15. Recognition Accuracy Results for this Study	96
Table 16. Symbol Definitions for this Section.....	97
Table 17. Video Datasets	143
Table 18. Overview of Video Processing Stages to Detect Edges and Binarize Videos.....	144
Table 19. Experimental Studies Described in this Section	153
Table 20. Code Similarity Decreases with Spatiotemporal Similarity of Moments.....	157
Table 21. The Code Selection Algorithm	159
Table 22. Reasoning Underlying the Global Spatiotemporal Familiarity Measure, G	162
Table 23. Reasoning Underlying the Computation of a Unit’s Local Support, V	163
Table 24. Variation of SISC Property with Parameter, π	168
Table 25. Top 8 IA Distributions Ranked by Average RA%	196
Table 26. Bottom 13 IA Distributions Ranked by Average RA%	196
Table 27. List of L2-Scale Simple Shape Features.....	201
Table 28. List of L3-Scale Simple Shape Features.....	203
Table 29. List of L4-Scale Simple Shape Features.....	204
Table 30. SOW Task Final Status.....	209
Table 31. Major Symbols in CSA Equations.....	218

1. ACKNOWLEDGEMENT

In addition to the authors of this report, I would like to acknowledge Harald Ruda, Nick Nowak and Anastasia Tyurina for the great supporting work they performed, which improved the quality of the research. I also would like to thank my two Program Managers, Dan Hammerstrom (Defense Advanced Research Projects Agency (DARPA)) and Kerry Hill (Air Force Research Laboratory (AFRL)), and their organizations for seeing the value of our research and for their strong support throughout the project.

2. SUMMARY

The overall goal of this two year project¹ was to investigate how representing information in the form of hierarchical sparse distributed representations (SDRs), a.k.a., sparse distributed codes (SDCs), yields extremely efficient solutions to spatiotemporal recognition, and more specifically, video event recognition, problems. As this is highly uncharted territory², this has been very much a basic research project. Though in the latter half of the project, we turned our focus more towards the applied research goal of achieving state-of-art (SOA) classification performance on benchmark video event recognition problems. We have carried out studies involving the Hollywood 2, KTH, and Weizmann data sets. In the final months of the performance period, we have focused almost exclusively on Weizmann in our effort to attain SOA performance. As of this date, the best performance of the specific model we are working with, Sparsey[®], is at 67% which is significantly below SOA, which is 100% (chance = 10%). However, our results so far should be viewed in light of the following critical points:

1. Sparsey's train time on Weizmann is ~3.5 minutes, with no machine parallelism whatsoever. An augmented Weizmann set (90 original snippets + 5 noisy versions of each of the original for a total of 540 snippets, are presented once each). Precise training times on Weizmann are not reported for any of the SOA-achieving models, but many use gradient-based learning and machine parallelism and likely have considerably longer training times. This raises a crucial point of distinction: Sparsey is leveraging *algorithmic* parallelism whereas most Deep Learning approaches have relied heavily on leveraging *machine* parallelism, specifically graphics processing units (GPUs). While machine parallelism is clearly essential, providing many orders of magnitude speedup, we argue in Section 4.8 that algorithmic parallelism, which is an orthogonal to and can be leveraged along with machine parallelism), actually yields even greater speedup.
2. The runtime of Sparsey's learning/inference algorithm depends only on fixed quantities, dominated by the number of weights, and therefore remains fixed for the life of the system. Regardless of how many sequences (snippets) have been stored, the time to learn the next snippet remains constant, as does the time to find the closest-matching stored snippet. We know of no other model, in particular the Deep Learning models, for which this claim has been made.
3. Sparsey uses extremely simple input features, binary pixels, as opposed to the more complex/informative features used by virtually all other systems, e.g., Histogram of Oriented Gradient (HOG), Histogram of Optical Flow (HOF), Motion Boundary Histograms (MBHs), and other such features. We believe it is likely that we could boost performance considerably by also using such features or by adding such features to our edge features. In fact, we did conduct a series of HOF-based studies but, for several reasons we did not attain high recognition accuracy and decided to focus back on using only our primitive edge/pixel features.

¹ Two NCTE's increased it to 2.5 years

² Other than Rinkus (2005, 2014), there are no other published reports describing hierarchical models that use SDR at multiple levels solving problems of any kind, let alone complex sequence recognition with time-warping and video event recognition.

4. An individual Sparsey module, which we call a macrocolumn, or "mac", since we view it as analogous to the cortical macrocolumn, has 100s of parameters (probably reducible to a few tens of meta-parameters). Thus far, we have done a very small amount of parameter searching and in fact, we still have a great deal to learn about the optimal relations between parameters both within any individual mac and between macs across levels. Thus, we are quite confident in being able to boost performance into the SOA range, i.e., 90-100%, relatively quickly.
5. The highly modular structure of the subject system, Sparsey, the local (in space and time) nature of its core algorithm, and the very low numerical precision it requires (binary neurons, 7-bit or less weights) all make it ideally suited to parallelization and novel low-power computing technologies.

In view of points 3 and 4 above, we have complete confidence that Sparsey will attain SOA classification accuracy on Weizmann and other video benchmarks in the very near future. We will continue on that research path and, given the technology's very high potential payoff, we will seek follow-on funding via appropriate channels.

We also anticipate significant transfer learning as we extend Sparsey to additional datasets. After all, all mid-level visual events (features) from essentially any natural visual domain can be composed, to reasonably good approximation, from a relatively small set of lower-level visual events. Similarly, all high-level visual events (again, from essentially any domain) can be composed from a relatively small set of mid-level visual events. As low- and mid-level features are learned (on the basis of the initial datasets presented to the model) at intermediate levels, additional novel higher-level events, e.g., from different datasets, will be learned progressively more quickly (because to a large extent, they are just novel spatiotemporal compositions of already-learned component features, and thus relatively little new learning is needed).

We believe our research has established the feasibility of hierarchical sparse distributed representations for the learning, recognition, prediction, of arbitrary spatiotemporal patterns (high-dimensional multivariate time series). Along the way, we have discovered a number of core broad design principles for the architecture and dynamics of such networks, described in Section 3.

3. INTRODUCTION

The biological brain, and the human brain in particular, remains the most advanced information processing device known. The remarkable structural homogeneity across the entire neocortical sheet suggests a core computational module, i.e., a “canonical cortical microcircuit”, operating similarly in all regions (Douglas, Martin et al. 1989, Douglas and Martin 2004, Rinkus 2010). The overarching rationale for our research is therefore that if we want to build computers that process information as well as humans, then we should understand the detailed structure and operation of said canonical cortical algorithm/circuit.

Based on a large body of evidence, we identify the canonical module with the cortical “macrocolumn” (a.k.a. “hypercolumn” in early visual cortex, or “barrel”-related volumes in rat/mouse primary somatosensory cortex). We’ll use “mac” as the generic name of the cortex’s core computational model and we’ll define it operationally as a volume of cortex, ~200-500 μm diameter, which operates autonomously and crucially, within which *SDRs*—i.e., small *sets* of the mac’s total pool of neurons—come to represent global input patterns experienced by the mac. The mac’s essential operations can be viewed as:

- a) *storage* (learning) of spatial/spatiotemporal input patterns, and
- b) *retrieval* of the closest matching, or most relevant, stored patterns.

As in most modern machine learning / pattern recognition models, the mac’s retrieval operation, which includes recognition, recall, and prediction, is construed / modeled as probabilistic inference. Our model of the canonical cortical algorithm, and thus of the operation of the mac, is known as the *Code Selection Algorithm* (CSA) and the overall theory, e.g., hierarchical networks consisting of large numbers of macs and possibly processing information from multiple sensory modalities, as Sparsey[®].

We emphasize at the outset that the spatiotemporal pattern learning/recognition model, Sparsey (formerly, TEMECOR), approaches learning, memory and recognition of patterns (information) in a fundamentally different manner than mainstream machine learning approaches. In the mainstream, usually task-driven, approach, the items of the training set, which are samples from a typically very high dimensional input space, are presented numerous times to the model and the model’s parameters (e.g., called *synaptic weights*) are moved in small increments up (or down) the gradient of some objective function, e.g., mean square error (MSE), so as to minimize classification error over the training set. The problem is that when the objective function is highly nonlinear, or more specifically, as described in (Bengio 2007), when the second derivative changes sign many times, following the gradient can become extremely slow or fail altogether. This is at the core of the reason why the mainstream, utilitarian view of learning, memory and recognition, i.e., of “cognition”, has failed to achieve truly human-like artificial intelligence. Can we do better by emulating biology?

We begin by noting a key difference between the mainstream, utilitarian view of learning, memory and recognition, i.e., of “cognition”, and biological, specifically human, cognition. In the former approach, there is no explicit need to learn/remember the details of the *individual* training items as long as overall classification performance is acceptable. Consequently,

traditional pattern recognition approaches generally *do not* explicitly remember these details: they “remember” only the parameters necessary to perform well with respect to the objective function.

In contrast, clearly humans generally *do* retain, sometimes for entire lifetimes, details of inputs that are not related to any particular task. In psychology, this type of detailed memory of specific inputs, e.g., specific spatiotemporal events experienced, has been referred to as *episodic* memory (Tulving 1972). We emphasize that by definition, single-trial (or at least, very-few-trial) learning is central to episodic memory: the types of events/episodes most germane here are unique events, e.g., your 5th birthday, your wedding day, the day Kennedy was shot, etc., which happen only once. Episodic memory is generally contrasted with *semantic* memory, which is memory for the meanings, i.e., classes, of objects/events, and of the features of objects/events, which are themselves viewable as just lower-level objects/events. Semantic memory therefore generally corresponds to the type of information learned/used in the mainstream approaches to pattern recognition, machine learning, and cognitive modeling. [Note that the idea that retaining memories of the individual items experienced is important in enabling a system to learn highly nonlinear classifications of the data is beginning to appear in the traditional machine learning research thread (Bengio, Courville et al. 2012)].

We ask: Is it possible that a system designed explicitly to retain as much information as possible about the specific inputs experienced, can simultaneously, as a by-product, learn/construct computationally efficient (from both a time and space perspective) representations of the, perhaps highly nonlinear, class structure, of input domains, in particular spatiotemporal domains, such as video? Sparsey, constitutes a positive answer to that question. As noted above, we have not yet shown SOA levels of performance on event classification. However, 67% is not too far short. Moreover, its time efficiency for both learning (storage) and closest-match retrieval (recognition) exceeds SOA. Specifically, for any particular model instance, both the time it takes to store a new input item (either spatial or spatiotemporal) and the time it takes to retrieve the closest-matching item (or in other words, recognize a novel item) *remains constant over the life of the system*. No other competing model has this time efficiency, including Semantic Hashing (Salakhutdinov and Hinton 2009) and Locality-Sensitive Hashing (LSH) (Andoni and Indyk 2008), for both of which the time it takes to learn new items increases with number of items. The key ingredient that makes this performance possible is the use of SDRs.

The switch from the mainstream approach to modeling cognition in which classification/recognition performance (semantic memory) is primary and simply storing (and later being able to remember) details of *individual* inputs experienced (episodic memory) is irrelevant to an approach in which episodic memory is primary and semantic memory is a by-product constitutes a *sea-change*. The idea of explaining semantic memory as being based on memories of individual items has been investigated for many years in the field of psychology (Hintzman 1986, Brooks 1987, Kruschke 1992, Whittlesea and Dorken 1993, Vokey and Brooks 1994). However, these theories and mechanisms have not been shown to be particularly efficient from a computational standpoint, most likely due primarily to the fact that these other models use localist representations rather than SDRs.

In addition, the experimental methods of neuroscience have matured enough so that we can now literally see memory traces (albeit, still rather coarsely) that very recently formed, recurring in the hours, days and weeks after their formation (Wilson and McNaughton 1994, Ji and Wilson 2007, Jafarpour, Fuentemilla et al. 2014). Hebb described the process of cognition in terms of “phase sequences”, essentially causal chains, of cell assemblies, i.e., SDRs (Hebb 1949). Luczak, McNaughton et al. (2015) provide further evidence for this view, though allowing some variability in the precise order of cell assemblies. Additional recent support consistent with this view comes from the Olshausen Lab [“Receptive field models don’t fail to predict responses of V1 neurons to natural movies”](#), Buszaki ([“neural syntax”](#) paper), and others. [Rafael Yuste’s Lab](#) is also directly focused on the problem of understanding how sparse distributed codes (they use “ensembles”) are learned and how they represent information in the brain. [This recent PNAS paper](#) strongly supports what we are seeing in our Sparsey simulations.

We are confident that as methods continue to mature, there will be increasing evidence that the primary mode of operation of cortex, and more specifically, each macrocolumn, is rapid (even single-trial) storage of novel inputs in the form of SDRs and immediate reactivation of those SDRs when future inputs are sufficiently similar to those originally stored.

3.1 The First Key: Sparse Distributed Representations

With respect to information processing, “intelligence” connotes two main functionalities:

- a) *Learning*, i.e., the ability to build up, via both unsupervised and supervised methods, representations of objects and relationships in the input domain; and
- b) Human-like *reasoning*, i.e., the ability to do logical/probabilistic inference/prediction involving the represented objects/relationships.

Formally, we consider a representation to include a set of *representational units* (RUs) for representing domain objects, both spatial and spatiotemporal, and a set of parameters, called *weights*, for representing relationships between objects. Representations may vary in the nature of the mapping from domain objects to RUs. In a *localist* representation, objects are represented by individual RUs, in 1-to-1 fashion. These are also referred to as *symbolic* representations since the representees of even the lowest-scale representations in the system, i.e., single RUs, are macroscopic domain entities, i.e., entities to which symbols (names) are typically attached. In contrast, in a *distributed* (or *sub-symbolic*) representation, all representees, even the smallest domain objects/events to which names are typically given, e.g., an oriented “edge” contour in a tiny region of visual space, are represented by *patterns of activity over multiple* RUs. Our proposed research will center on a particular kind of distributed representation scheme, termed SDR, in which the RUs are binary and objects are represented by (relatively) small *sets* of RUs chosen from a much larger overall population of RUs.³ We describe the specific SDR formalism used by Sparsey in Section 3.

³ Our use of “sparse distributed” differs from its mainstream usage in computational neuroscience, popularized by Olshausen and Field, in which “sparse” means that the set of features comprising a representational basis (lexicon) is small compared to set of all possible features definable on the input space, and “distributed” means that representation of any particular input is generally a *composition of multiple* features. However, to my knowledge, all models described as sparse and distributed, in this mainstream sense, are in fact *localist*, i.e., each individual

The vast majority of artificial intelligence (AI) and machine intelligence (MI) models and undoubtedly, virtually all commercial software systems use localist representations. Most neuromorphic AI/MI models—i.e., those whose learning schemes are characterized in terms of gradient descent/ascent or statistical sampling—have used densely or fully distributed representations. Only a tiny fraction of AI/MI research to date has focused on SDR, e.g., (Kanerva 1988, Moll and Miikkulainen 1995, Rinkus 1996, Rachkovskij and Kussul 2001, Hecht-Nielsen 2005). Nevertheless, we must emphasize that there is considerable evidence that the only truly intelligent system known, the biological brain, uses sparse distributed representations throughout cortex.

3.2 The Second Key: Hierarchy

Representations may also vary in the pattern of connections (weights) amongst the RUs. In particular, the RUs may be divided into any number of *levels* such that the RUs in any level connect directly only with RUs in their own and adjacent levels. Multi-level representations are referred to as *hierarchical*. Hierarchy per se has been used to improve information storage/processing efficiency since time immemorial, from the game of “20 Questions” to computer algorithms and architectures. And, it is explicitly used in the majority of AI/MI models, e.g., semantic networks, Bayesian belief nets. It is therefore uncontentious to claim that *optimizing the usage of hierarchy* should be a key to achieving scalable MI, where by this we mean building domain models:

- a) that remain tractable while scaling to ~8-10 levels, and beyond, but more importantly,
- b) in which the components of hierarchical representations of high-level objects, e.g., airplane, correspond to *natural* parts of those hierarchical objects, e.g., wing, fuselage, tail, horizontal stabilizer, engine, etc., i.e., systems which learn representations that “cut nature at its joints”.

We emphasize that for a large fraction of the sampling/gradient-based neuromorphic model classes alluded to above, extension beyond three or four levels had remained impractical until recently (Hinton, Osindero et al. 2006). However, even with the recent advances in training deep belief nets, there has yet been very little compelling demonstration of the second aspect of optimizing the usage of hierarchy (point b, above), i.e., of hierarchical representations being learned, whose components correspond to *natural* parts of the overall objects. And, this is especially true in the spatiotemporal domain, i.e., activity/event recognition. Yet, demonstration of such is needed to provide a more immediate and compelling bridge to top-down “AI” approaches, which assume *a priori*, compositional, grammatical, symbolic knowledge representations and rules of inference. We believe that this particular “cutting nature at its joints” criterion is essential if we are to understand, interact with, and trust applications charged with monitoring, mining, and predicting over heterogeneous databases of massive scale.

feature is represented by a single unit. In contrast, in our approach, representations of all objects/events/features, at all levels, are relatively small (*sub*)sets of RUs from a much larger population. Note: the two usages are completely compatible and in fact, our SDR-based model is “sparse” and “distributed” in both senses.

3.3 Putting the Two Keys Together

The hierarchical organization of visual cortex is captured in many biologically inspired computational vision models with the general idea being that progressively larger scale (both spatially and temporally) and more complex visual features are represented in progressively higher areas (Riesenhuber and Poggio 1999, Serre, Kouh et al. 2005). Our cortical model, Sparsey, is hierarchical as well, but as noted above, a crucial, in fact, the most crucial difference between Sparsey and most other biologically inspired vision models is that Sparsey encodes information at all levels of the hierarchy, and in every mac at every level, with SDCs. This stands in contrast to models that use *localist representations*, e.g., all published versions of the HMAX family of models, e.g., (Murray and Kreutz-Delgado 2007, Serre, Kreiman et al. 2007) and other cortically-inspired hierarchical models (Kouh and Poggio 2008, Litvak and Ullman 2009, Jitsev 2010) and the majority of graphical probability-based models (e.g., hidden Markov models, Bayesian nets, dynamic Bayesian nets). There are several other models for which SDC is central, e.g., SDM (Kanerva 1988, Kanerva 1994, Jockel 2009, Kanerva 2009), Convergence-Zone Memory (Moll and Miikkulainen 1997), Associative-Projective Neural Networks (Rachkovskij 2001, Rachkovskij and Kussul 2001), Cogent Confabulation (Hecht-Nielsen 2005), Valiant's "positive shared" representations (Valiant 2006, Feldman and Valiant 2009), and Numenta's Grok (described in Numenta white papers). However, none of these models has been substantially elaborated or demonstrated in an explicitly hierarchical architecture and most have not been substantially elaborated for the spatiotemporal case.

Figure 1 illustrates the difference between a localist, e.g., an HMAX-like, model and the SDC-based Sparsey model. The input level (analogous to thalamus) is the same in both cases: each small gray/red hexagon in the input level represents the aperture (U receptive field (U-RF)) of a single V1 mac (gray/red hexagon). In Figure 1a, the representation used in each mac (at all levels) is *localist*, i.e., each feature is represented by a single cell and at any one time, only one cell (feature) is active (red) in any given mac (here the cell is depicted with an icon representing the feature it represents). In contrast, in Figure 1b, any particular feature is represented by a *set* of co-active cells (red), one in each of a mac's minicolumns: compare the two macs at lower left of Figure 1a with the corresponding macs in Figure 1b (blue and brown arrows). Any given cell will generally participate in the codes of many different features. A yellow call-out shows codes for other features stored in the mac, besides the feature that is currently active. If you look closely, you can see that for some macs, some cells are active in more than one of the codes.

Looking at Figure 1a, adapted from Serre, Kouh et al. (2005), one can see the basic principle of hierarchical compositionality in action. The two neighboring apertures (pink) over the dog's nose lead to activation of cells representing a vertical and a horizontal feature in neighboring V1 macs. Due to the convergence/divergence of U-projections to V2, both of these cells project to the cells in the left-hand V2 mac. Each of these cells projects to multiple cells in that V2 mac, however, only the red (active) cell representing an "upper left corner" feature, is maximally activated by the conjunction of these two V1 features. Similarly, the U-signals from the cell representing the "diagonal" feature active in the right-hand V1 mac will combine with signals representing features in nearby apertures to activate the appropriate higher-level feature in the V2 mac whose U-RF includes these apertures (small dashed circles in the input level). Note that some notion of competition (e.g., the "max" operation in HMAX models) operates amongst the cells of a mac such that at any one time, only one cell (one feature) can be active.

We underscore that in Figure 1, we depict simple (solid border) and complex (dashed border) features within individual macs, implying that complex and simple features can compete with each other. We believe that the distinction between simple and complex features may be largely due to coarseness of older experimental methods (e.g., using synthetic low-dimensional stimuli): newer studies are revealing far more precise tuning functions (Nandy, Sharpee et al. 2013), including temporal context specificity, even as early as V1 (DeAngelis, Ohzawa et al. 1993, DeAngelis, Ghose et al. 1999), and in other modalities, somatosensory (Ramirez, Pnevmatikakis et al. 2014) and auditory (Theunissen and Elie 2014).

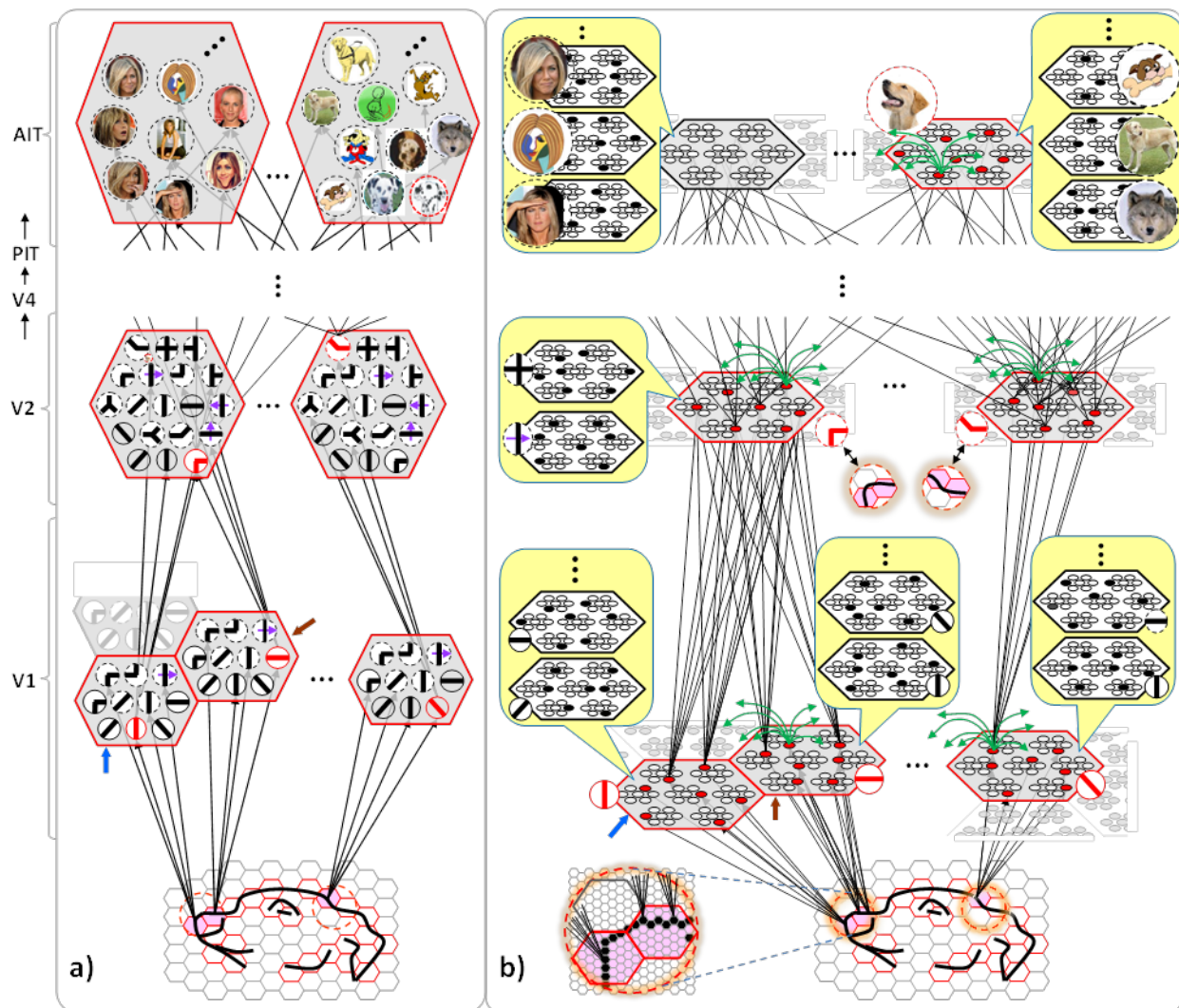


Figure 1: Comparison of Localist and SDC-based Hierarchical Vision Models

The same hierarchical compositional scheme as between V1 and V2 continues up the hierarchy (some levels not shown), causing activation of progressively higher-level features. At higher levels, we typically call them concepts, e.g., the visual concept of “Jennifer Aniston”, the visual concept of the class of dogs, the visual concept of a particular dog, etc. We show most of the features at higher levels with dashed outlines to indicate that they are *complex* features, i.e., features with particular, perhaps many, dimensions of invariance, most of which are learned

through experience. In Sparsey, the particular invariances are learned from scratch and will generally vary from one feature/concept to another, including within the same mac. The particular features shown in the different macs in this example are purely notional: it is the overall hierarchical compositionality principle that is important, not the particular features shown, nor the particular cortical regions in which they are shown.

The hierarchical compositional process described above in the context of the localist model of Figure 1a applies to the SDC-based model in Figure 1b as well. However, features/concepts are now represented by sets of cells rather than single cells. Thus, the vertical and horizontal features forming part of the dog's nose are represented with SDCs in their respective V1 macs (blue and brown arrows, respectively), rather than with single cells. The U-signals propagating from these two V1 macs converge on the cells of the left-hand V2 mac and combine, via Sparsey's CSA, to activate the SDC representing the "corner" feature, and similarly on up the hierarchy. Each of the orange outlined insets at V2 shows the input level aperture of the corresponding mac, emphasizing the idea that the precise input pattern is mapped into the closest-matching stored feature, in this example, a "upper left 90° corner" at left and a "NNE-pointing 135° angle" at right. The inset at bottom of Figure 1b zooms in to show that the U-signals to V1 arise from individual pixels of the apertures (which would correspond to individual LGN projection cells).

In the past, IT cells have generally been depicted as being *narrowly selective* to particular objects (Desimone, Albright et al. 1984, Kreiman, Hung et al. 2006, Kiani, Esteky et al. 2007, Rust and DiCarlo 2010). However, as DiCarlo, Zoccolan et al. (2012) point out, the data overwhelmingly support the view of individual IT cells as having a "diversity of selectivity"; that is, individual IT cells generally respond to many different objects and in that sense are much more broadly tuned. This diversity is notionally suggested in Figure 1b and Figure 2 in that individual cells are seen to participate in multiple SDCs representing different images/concepts. However, the particular input (stimulus) dimensions for which any given cell ultimately demonstrates some degree of invariance is not prescribed a priori. Rather they emerge essentially idiosyncratically over the history of a cell's inclusions in SDCs of particular experienced moments. Thus, the dimensions of invariance in the tuning functions of even immediately neighboring cells may generally end up quite different.

Figure 2 embellishes the scheme shown in Figure 1b and (turning it sideways) casts it onto the physical brain. We add paths from V1 and V2 to a MT representation as well. We add a notional PFC representation in which a higher-level concept involving the dog, i.e., the fact that it is being walked, is active. We show a more complete tiling of macs at V1 than in Figure 1b to emphasize that only V1 macs that have a sufficient fraction of active pixels, e.g., an edge contour, in their aperture become active (pink). In general, we expect the fraction of active macs to decrease with level. As this and prior figures suggest, we currently model the macs as having no overlap with each other (i.e., they tile the local region), though their RFs [as well as their *projective fields* (PFs)] can overlap. However, we expect that in the real brain, macs can physically overlap. That is, any given minicolumn could be contained in multiple overlapping macs, where only one of those macs can be active at any given moment. The degree of overlap could vary by region, possibly generally increasing anteriorly. If so, then this would partially explain (in conjunction with the extremely limited view of population activity that single/few-

unit electrophysiology has provided through most of the history of neuroscience) why there has been little evidence thus far for macs in more frontal regions.

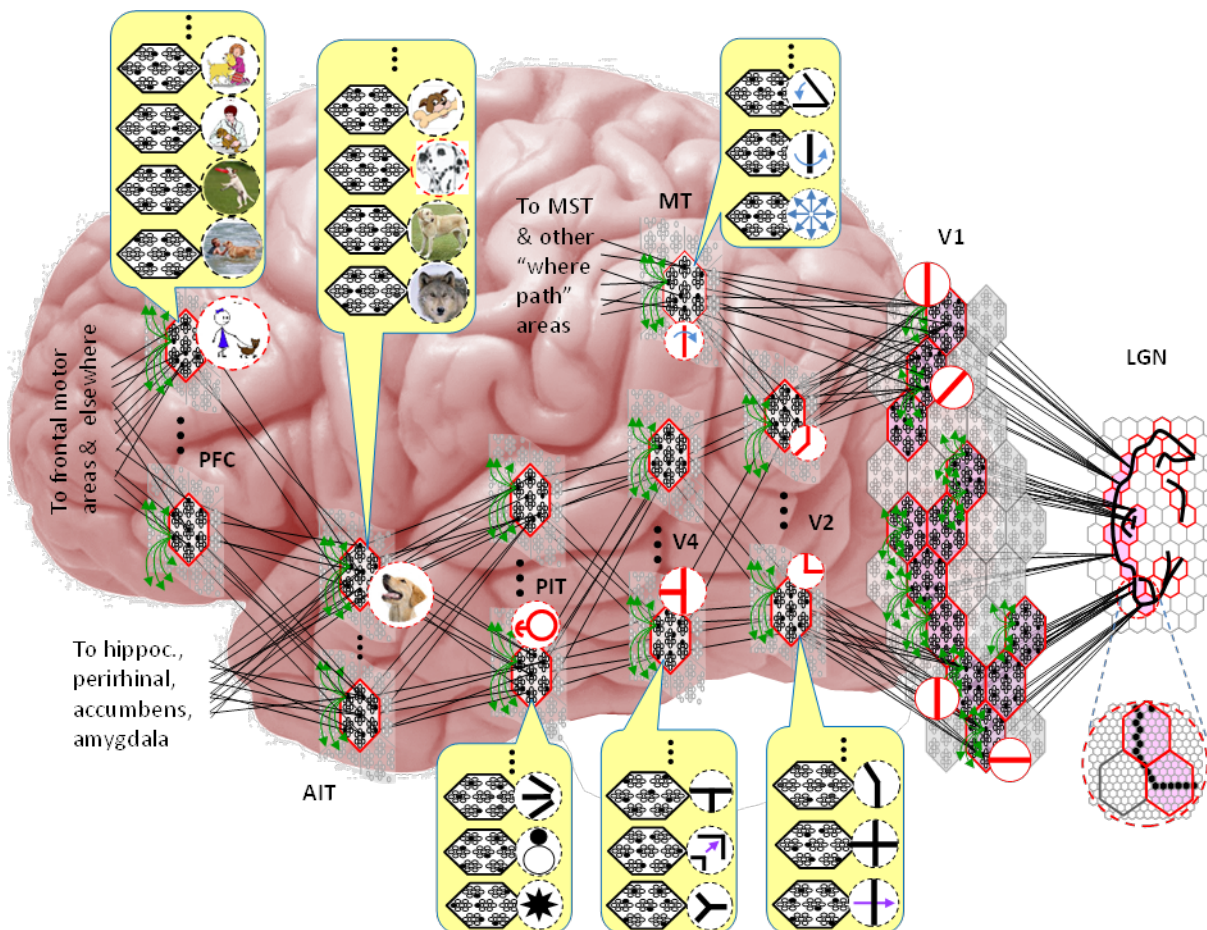


Figure 2: Notional Mapping of Sparsey to Brain

3.4 Final State of the Research

As mentioned earlier, our highest accuracy achieved thus far on a benchmark video event recognition dataset, Weizmann, is 67%, which trained in 3.5 minutes. The overall experimental protocol was:

1. Unsupervised pre-training on augmented training set
2. Supervised leave one out (LOO) support vector machine (SVM) training
3. Test on the held-out snippets

To augment the data, we created additional training snippets by making five new 10% noisified versions of the each of the 90 original Weizmann snippets (Figure 4 shows an example of a 20% noisified frame). We call these noisy versions “cousins” and the group consisting of an original and its five cousins a “cousin-group”. We pre-trained will all 540 snippets (90 cousin groups x 6 cousins per group).

We then conducted the supervised LOO SVM training. Specifically, for each of the 540 training snippets, we hold out one of those snippets and its five cousins, training on the remaining 534 (89x6) snippets. In experimental condition 1 (Column 2 of Table 1), we test on the six (original + 5 cousins) held-out snippets. We then average over the 90 LOO iterations, i.e., over the $90 \times 6 = 540$ tested snippets. In experimental condition 2 (Column 3 of Table 1), we test only on the original snippet of each cousin-group; thus we present only the original 90 Weizmann snippets at test and average over them.

As Table 1 shows, our best result (67%) is constant across 5 orders of magnitude of the C parameter. We are somewhat surprised that we do so much better when the final testing is only on the 90 original snippets as opposed to on all 540 snippets (originals + augmented). We have double-checked the sanity of our methods, but we would like to further understand the underlying cause. As noted earlier, the unsupervised training takes 3.5 minutes on a single processor. The SVM takes an additional 2.5 minutes in the “Test on All” condition, but only a few seconds on the “Test only on Originals” condition. In any case, our use of this protocol involving the SVM is only temporary. Once the parameter tuning is completed, we will return to our original protocol where at test, the top-level codes directly drive the category nodes.

Table 1. Classification Results on Weizmann Data as Function of SVM Cost Parameter

C	Test on All	Test Only on Original Snippets.
100	0.50	0.67
10	0.50	0.67
1	0.50	0.67
0.1	0.50	0.67
0.01	0.50	0.67
0.001	0.47	0.60
0.0001	0.47	0.59
0.00001	0.47	0.60
0.000001	0.47	0.59
0.0000001	0.46	0.59

Figure 3 shows a picture of the model used in these studies on one particular frame of one snippet. L1 has 12x18 macs and L2 had 6x9 macs. Example videos of the network in action can be seen at [this web page](#). It has approximately 7.5 million weights. The U-RFs at L1 of three L2 macs (green outline) are shown in cyan (active L1 macs within cyan patches are purple). Two U-RFs are overlapped and their borders cannot be discerned. The U-RFs in L0 of several active L1 macs are also shown; they are the darker cyan patches in L0. The lighter cyan patches in L0 are the L0 U-RFs of the L2 macs (i.e., as mediated by the L1 macs). The black pixels in the L1 mac U-RFs are the active pixels which are causing those L1 macs to activate. The purple L1 macs in the L1 mac U-RFs are the active macs which are causing the L2 macs to activate.

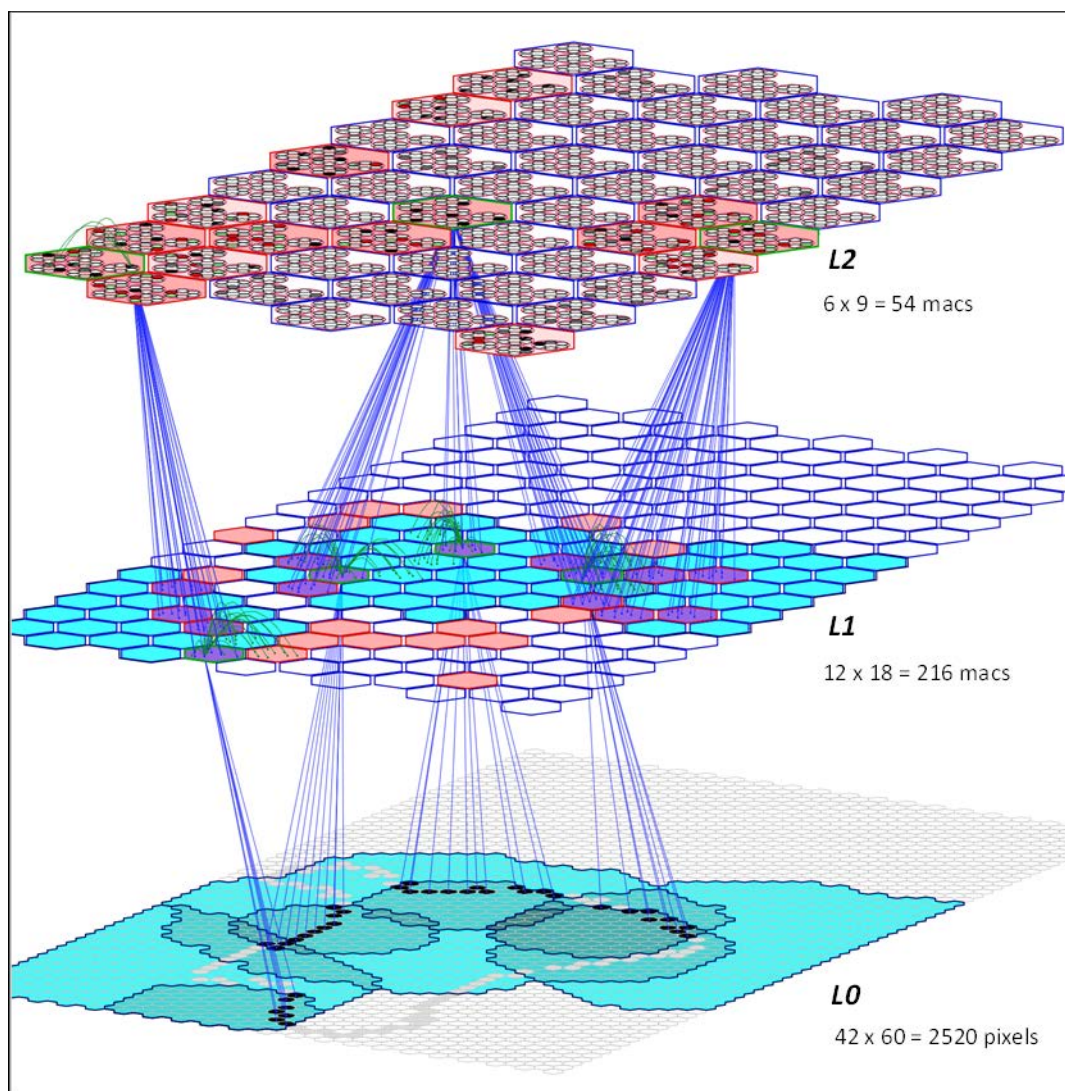


Figure 3: Snapshot of Best Performing Model in Study

4. METHODS, ASSUMPTIONS, AND PROCEDURES

4.1 Overview of Model Architecture

In contrast, because SDCs physically overlap, if one particular SDC (and thus, the hypothesis that it represents) is *fully* active in a mac, i.e., if all Q of that code's cells are active, *then all other codes (and thus, their associated hypotheses) stored in that mac are also simultaneously physically partially active in proportion to the size of their intersections with the single fully active code*. Furthermore, if the process/algorithm that assigns the codes to inputs has enforced the *similar-inputs-to-similar-codes* (SISC) property, then all stored inputs (hypotheses) are active with strength in descending order of similarity to the fully active hypothesis. We assume that more similar inputs generally reflect more similar world states and that world state similarity correlates with likelihood. In this case, the single fully active code also physically functions as the *full likelihood distribution over all SDCs (hypotheses) stored in a mac*. Figure 4 illustrates this concept. We show five hypothetical SDCs, denoted with $\phi()$, for five input items, A-E (the actual input items are not shown here), which have been stored in the mac shown. At right, we show the decreasing intersections of the codes with $\phi(A)$. Thus, when code $\phi(A)$ is (fully) active, $\phi(B)$ is 4/7 active, $\phi(C)$ is 3/7 active, etc. Since cells representing *all* of these hypotheses, not just the most likely hypothesis, A, actually spike, it follows that *all of these hypotheses physically influence the next time step's decision processes*, i.e., the resulting likelihood distributions, active on the next time step in the same and all downstream macs. At bottom of the figure, we show the activation strength distribution over all five codes (stored hypotheses), when each of the five codes is fully active. If SISC was enforced when these codes were assigned (learned), then these distributions are interpretable as likelihood distributions.

We believe this difference to be fundamentally important. In particular, it means that performing a single execution of the *fixed-time* CSA transmits the influence of *every* represented hypothesis, regardless of how strongly active a hypothesis is, to every hypothesis represented in downstream macs. We emphasize that the representation of a hypothesis's probability (or likelihood) in our model—i.e., as the fraction of a given hypothesis's full code (of Q cells) that is active—differs fundamentally from existing representations in which single neurons encode such probabilities in their strengths of activation (e.g., firing rates) as described in the recent review of (Pouget, Beck et al. 2013).

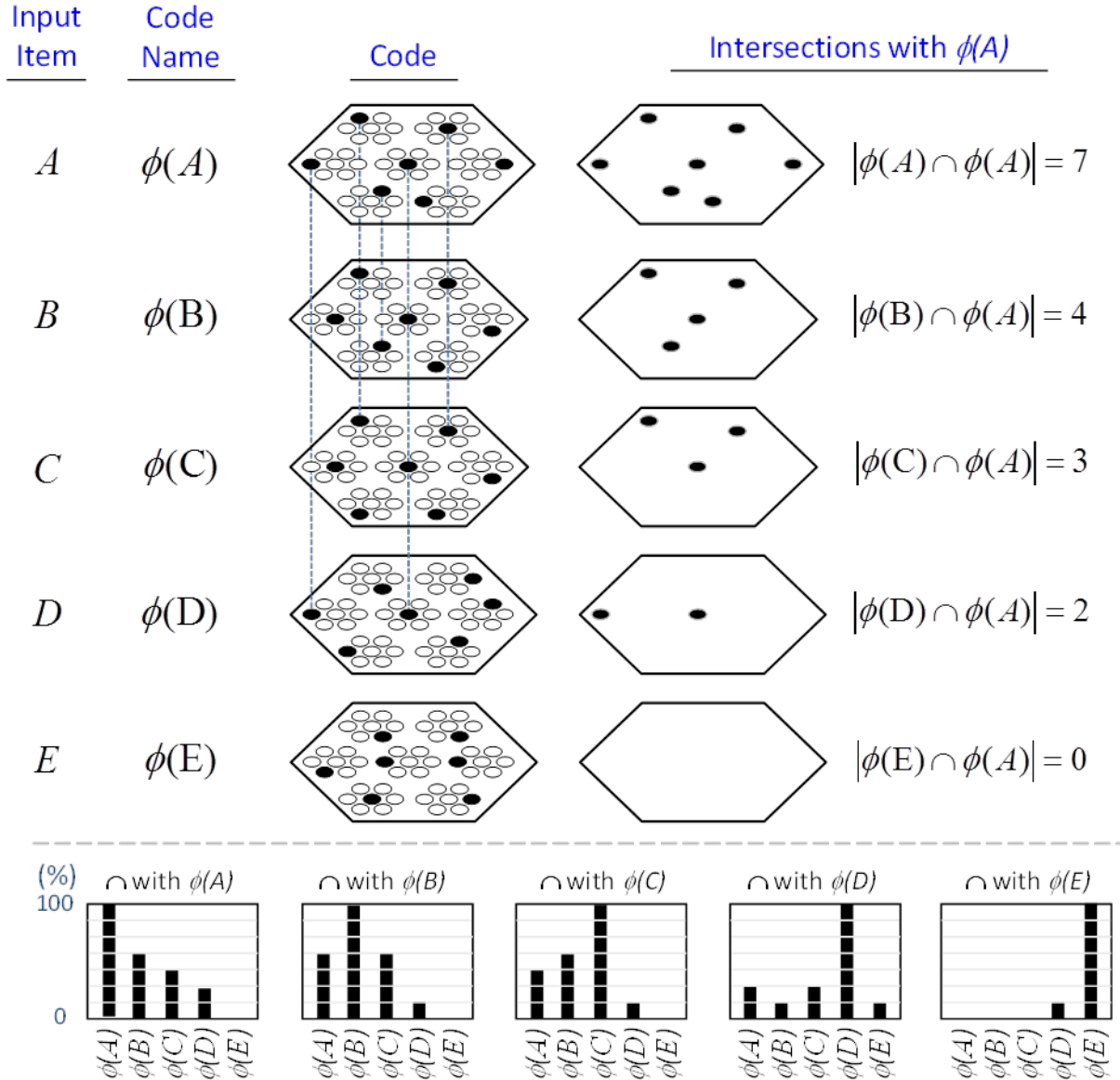


Figure 4: The Currently Active Mac Code Simultaneously Physically Functions as the Entire Likelihood Distribution over all Hypotheses Stored in the Mac

4.2 The Algorithm

During learning, Sparsey’s core algorithm, the CSA, operates on every time step (frame) in every mac of every level, resulting in activation of a set of cells (an SDC) in the mac. The CSA can also be used, with one major variation, during retrieval (recognition). However, there is a much simpler retrieval algorithm, essentially just the first few steps of the CSA, which is preferable if the system “knows” that it is in retrieval mode. Note that this is not the natural condition for autonomous systems: in general, the system must be able to decide for itself, on a frame-by-frame basis, whether it needs to be in learning mode (if, and to what extent, the input is novel) or retrieval mode (if the input is completely familiar). We first describe the CSA’s learning mode,

then its variation for retrieval, then it's much simpler retrieval mode. The symbols used throughout this report are defined in Table 31.

4.2.1. CSA: Learning Mode

The overall goal of the CSA when in learning mode is to assign codes to a mac's inputs in adherence with the SISC property, i.e., more similar overall inputs to a mac are mapped to more highly intersecting SDCs. With respect to each of a mac's individual afferent RFs, U, H, and D, the similarity metric is extremely primitive: the similarity of two patterns in an afferent RF is simply an increasing function of *the number of features* in common between the two patterns, thus embodying only what Bengio, Courville et al. (2012) refer to as the weakest of priors, the *smoothness* prior. However, the CSA multiplicatively combines these component similarity measures and, because the H and D signals carry temporal information reflecting the history of the sequence being processed, the CSA implements a spatiotemporal similarity metric. Nevertheless, the ability to learn arbitrarily complex *nonlinear* similarity metrics (i.e., category boundaries, or invariances), requires a hierarchical network of macs and the ability for an individual SDC, e.g., active in one mac, to associate with multiple (perhaps arbitrarily different) SDCs in one or more other macs.

The CSA has 12 steps which can be broken into two phases. Phase 1 (Steps 1-7) culminates in computation of the *familiarity*, G (normalized to $[0,1]$), of the overall (H, U, and D) input to the mac as a whole, i.e., G is a function of the *global* state of the mac. To first approximation, G is the similarity of the current overall input to the closest-matching previously stored (learned) overall input. As we will see, computing G involves a round of deterministic (hard max) competition resulting in one winning cell in each of the Q competitive modules (CMs). In Phase 2 (Steps 8-12), the activation function of the cells is modified based on G and a second round of competition occurs, resulting in the final set of Q winners, i.e., the activated code in the mac on the current time step. The second round of competition is *probabilistic* (soft max), i.e., the winner in each CM is chosen as a draw from a probability distribution over the CM's K cells.

In neural terms, each of the CSA's two competitive rounds entail the principal cells in each CM integrating their inputs, engaging the local inhibitory circuitry, resulting in a single spiking winner. The difference is that the cell activation functions (F/I-curves) used during the second round of integration will generally be very different from those used during the first round. Broadly, the goal is as follows: as G approaches 1, make cells with larger inputs compared to others in the CM increasingly likely to win in the second round, whereas as G approaches 0, make all cells in a CM equally likely to win in the second round.

We now describe the steps of the CSA in learning mode. We will refer to the generic "circuit model" in Figure 5 in describing some of the steps. The figure has two internal levels with one small mac at each level, but the focus, in describing the algorithm, will be on the L1 mac, M_j^1 ,

highlighted in yellow. M_j^1 consists of $Q=4$ CMs, each with $K=3$ cells. Gray arrows represent the U-wts from the input level, L0, consisting of 12 binary pixels. Magenta arrows represent the D-wts from the L2 mac. Green lines depict a subset of the H-wts. The representation of where the different afferents arrive on the cells is not intended to be veridical. The depicted "Max"

operations are the hard max operations of CSA Step 7. The blue arrows portray the mac-global G -based modulation of the cellular V -to- ψ map (essentially, the F/I curve). The probabilistic draw operation is not explicitly depicted in this circuit model.

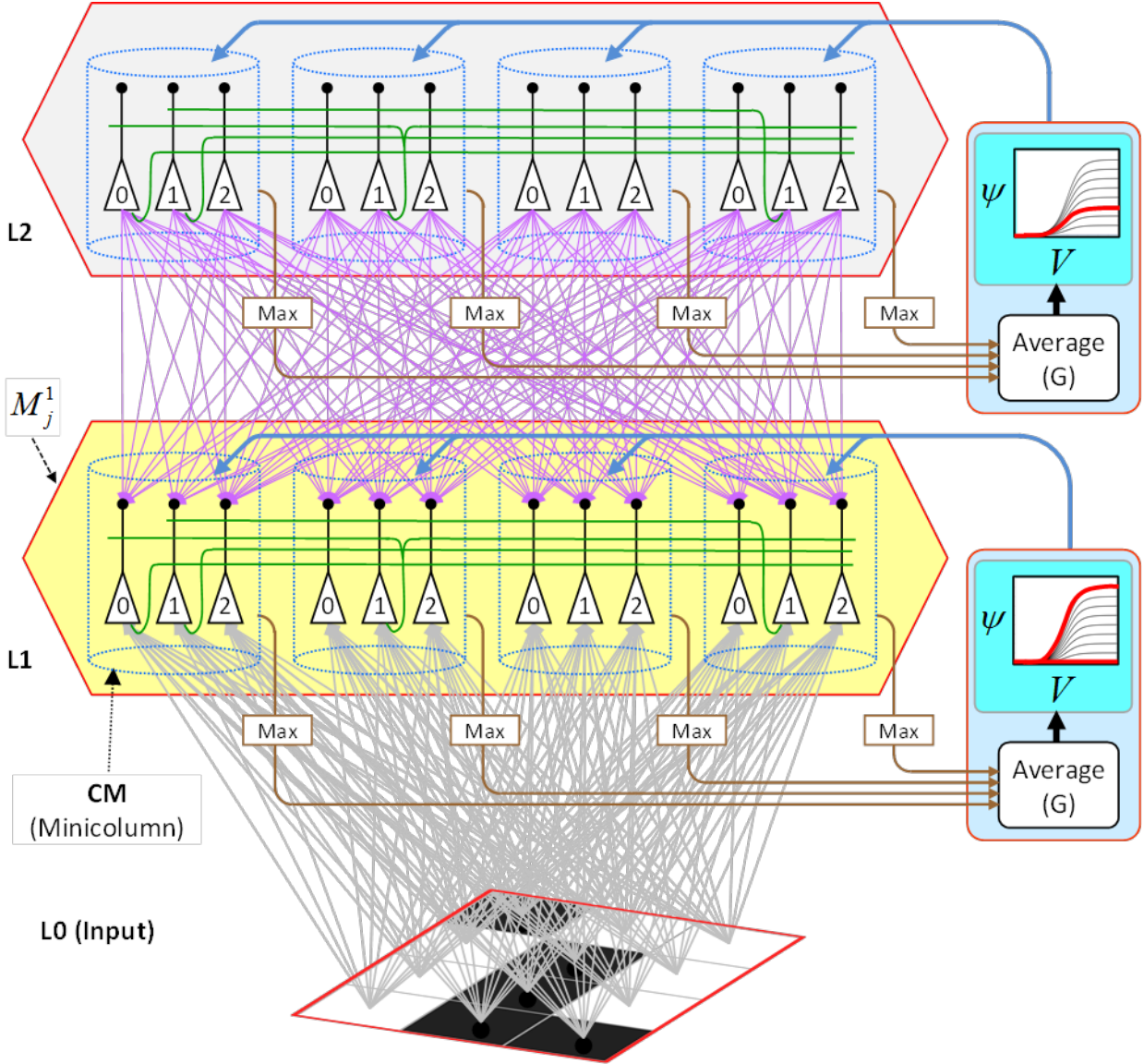


Figure 5: Generic “Circuit Model” for Reference in Describing Some Steps of the CSA

4.2.1.a. Step 1: Determine if the Mac will become Active

As shown in Eq. 1, during learning, a mac, m , becomes active if either of two conditions hold: a) if the number of active features in its U-RF, $\pi_U(m)$, is between π_U^- and π_U^+ ; or b) if it is already active but the number of frames that it has been on for, i.e., its *code age*, $\Upsilon(m)$, is less than its persistence, $\delta(m)$. That is, during learning, we want to ensure that codes remain on for their entire prescribed persistence durations. We currently have no conditions on the number of active features in the H and D RFs.

$$Active(m) = \begin{cases} true & Y(m) < \delta(m) \\ true & \pi_U^- \leq \pi_U(m) \leq \pi_U^+ \\ false & otherwise \end{cases} \quad (\text{Eq. 1})$$

4.2.1.b. Step 2: Compute Raw U, H, and D-Summations for each Cell, i, in the Mac

Every cell, i , in the mac computes its three weighted input summations, $u(i)$, as in Eq. 2a. RF_U is a synonym for U-RF. $a(j, t)$ is pre-synaptic cell j 's activation, which is binary, on the current frame. Note that the synapses are *effectively binary*. Although the weight range is $[0, 127]$, pre-post correlation causes a weight to increase immediately to $w_{\max} = 127$ and the asymptotic weight distribution will have a tight cluster around 0 (for weights that are effectively “0”) and around 127 (for weights that are effectively “1”). The learning policy and mechanics are described in Section 0. $F(\zeta(j, t))$ is a term needed to adjust the weights of afferent signals from cells in macs in which multiple competing hypotheses (MCHs) are active. If the number of MCHs (ζ) is small then we want to boost the weights of those signals, but if it gets too high, in which case we refer to the source mac as being *muddled*, those signals will generally only serve to decrease SNR in target macs and so we disregard them. Computing and dealing with MCHs is described in Steps 5 and 6. $h(i)$ and $d(i)$ are computed in analogous fashion (Eqs. 2b,c), with the slight change that H and D signals are modeled as originating from codes active on the previous time step ($t-1$).

$$u(i) = \sum_{j \in RF_U} a(j, t) \times F(\zeta(j, t)) \times w(j, i) \quad (\text{Eq. 2a})$$

$$h(i) = \sum_{j \in RF_H} a(j, t-1) \times F(\zeta(j, t-1)) \times w(j, i) \quad (\text{Eq. 2b})$$

$$d(i) = \sum_{j \in RF_D} a(j, t-1) \times F(\zeta(j, t-1)) \times w(j, i) \quad (\text{Eq. 2c})$$

4.2.1.c. Step 3: Normalize and Filter the Raw Summations

The summations, $u(i)$, $h(i)$, and $d(i)$, are normalized to $[0, 1]$ interval, yielding $U(i)$, $H(i)$, and $D(i)$. We explained above that a mac m only becomes active if the number of active features in its U-RF, $\pi_U(m)$, is between π_U^- and π_U^+ , referred to as the lower and upper *mac activation bounds*. Given our assumption that visual inputs to the model are filtered to single-pixel-wide edges and binarized, we expect relatively straight or low-curvature edges roughly spanning the diameter of an L0 aperture to occur rather frequently in natural imagery. Figure 6 shows two examples of such inputs, as frames of sequences, involving either only a single L0 aperture (panel a) or a region consisting of three L0 apertures, i.e., as might comprise the U-RFs of an L2 mac (panel b). The general problem, treated in this figure, is that the number of features present in a mac's U-RF, $\pi_U(m)$, may vary from one frame to the next. In Figure 6a, an edge rotates through the aperture over three time steps, but the number of active features (in this case, pixels) varies from one time step (moment) to the next. In order for the mac to be able to recognize the 5-pixel input ($T=1$) just as strongly as the 6 or 7-pixel inputs, the u-summations must be divided by 5. In Figure 6b, the U-RFs of macs at L2 and higher consist of an integer number of subjacent level macs, e.g., here, M_i^2 's U-RF consists of three L1 macs (blue border). Each active

mac in M_i^2 's U-RF represents one feature. As for panel a, the number of active features varies across moments, but in this case, the variation is in increments/decrements of Q synaptic inputs. Grayed-out apertures have too few active pixels for their associated L1 macs to become active.

Note that for macs at L2 and higher, the number of features present in an RF is the *number of active macs* in that RF, not the total number of active cells in that RF. The policy implemented in Sparsey is that inputs with different numbers of active features compete with each other on an equal footing. Thus, normalizers (denominators) in Eqs. 3a,b,c use the lower mac activation bound, π_U^- , π_H^- , and π_D^- . This necessitates hard limiting the maximum possible normalized value to 1, so that inputs with between π_U^- and π_U^+ active features yield normalized values confined to [0,1]. There is one additional nuance. As noted above, if a mac in m 's U-RF is muddled, then we disregard all signals from it, i.e., they are not included in the u -summations of m 's cells. However, since that mac is active, it will be included in the number of active features, $\pi_U(m)$. Thus, we should normalize by the number of active, *non-muddled* macs in m 's U-RF (not simply the number of active macs): we denote this value as π_U^* . Finally, note that when the afferent feature is represented by a mac, that feature is actually being represented by the simultaneous activation of, and thus, inputs from, Q cells; thus the denominator must be adjusted accordingly, i.e., multiplied by Q and by the maximum weight of a synapse, w_{\max} .

$$U(i) = \begin{cases} \max(1, u(i)/\pi_U^- \times w_{\max}) & L = 1 \\ \max(1, u(i)/\min(\pi_U^-, \pi_U^*) \times Q \times w_{\max}) & L > 1 \end{cases} \quad (\text{Eq. 3a})$$

$$H(i) = \max(1, h(i)/\min(\pi_H^-, \pi_H^*) \times Q \times w_{\max}) \quad (\text{Eq. 3b})$$

$$D(i) = \max(1, d(i)/\min(\pi_D^-, \pi_D^*) \times Q \times w_{\max}) \quad (\text{Eq. 3c})$$

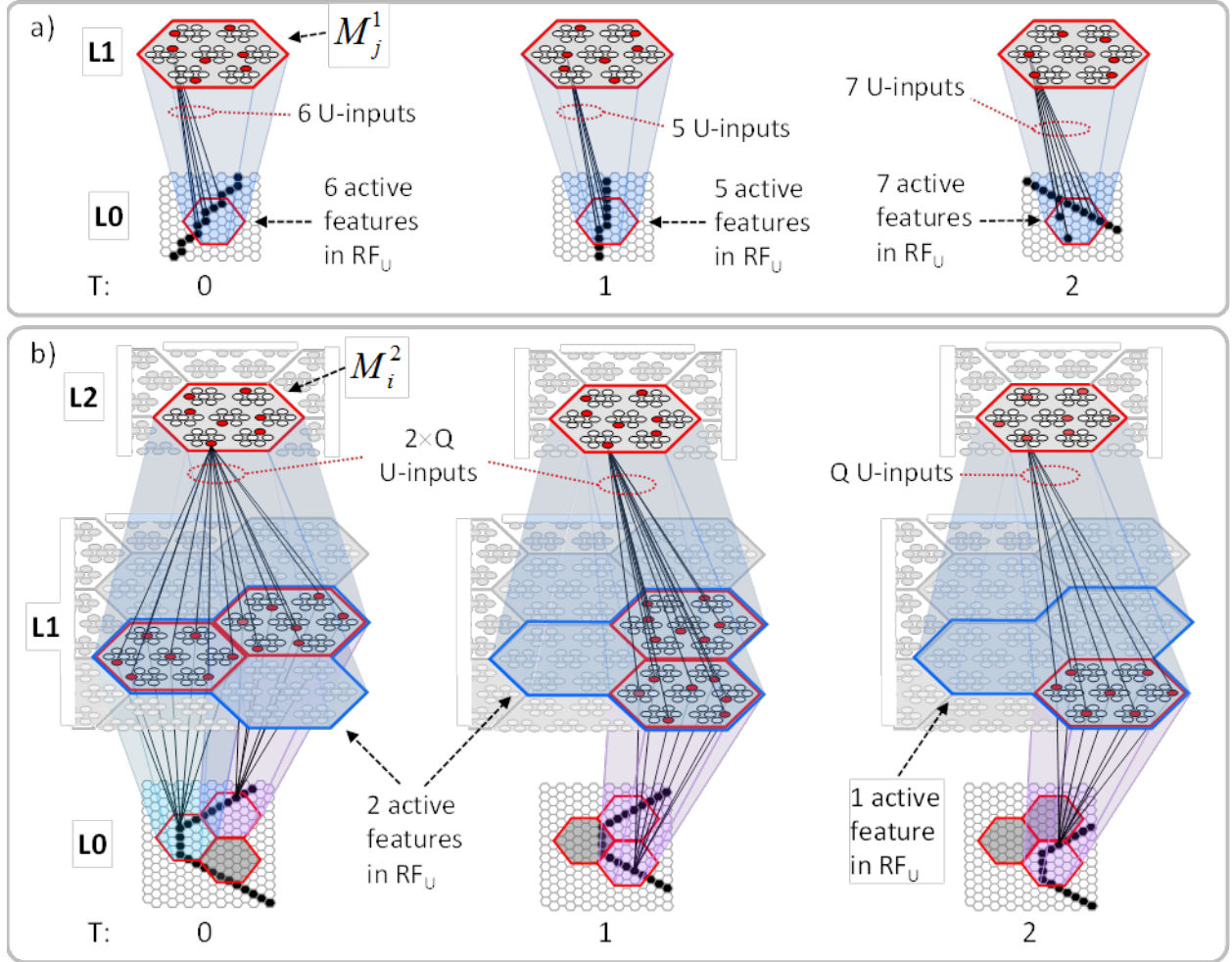


Figure 6: Mac Normalization Policy Handles Inputs with Varying Numbers of Features

4.2.1.d. Step 4: Compute Overall Local Support for Each Cell in the Mac

The overall *local* (to the individual cell) measure, $V(i)$, of evidence/support that cell i should be activated is computed by multiplying filtered versions of the normalized inputs as in Eq. 4. $V(i)$ can also be viewed as the normalized degree of match of cell i 's total afferent (including U, H, and D) synaptic weight vector to its total input pattern. We emphasize that the V measure is *not* a measure of support for a *single* hypothesis, since an individual cell does not represent a single hypothesis. Rather, in terms of hypotheses, $V(i)$ can be viewed as the local support for the *set* of hypotheses whose representations (codes) include cell i . The individual normalized summations are raised to powers (λ), which allows control of the relative sensitivities of V to the different input sources (U, H, and D). Currently, the U-sensitivity parameter, λ_U , varies with time (index of frame with respect to beginning of sequence). We will add time-dependence to the H and D sensitivity parameters as well and explore the space of policies regarding these schedules in the future. In general terms, these parameters (along with many others) influence the shapes of the boundaries of the categories learned by a mac.

$$V(i) = \begin{cases} H(i)^{\lambda_H} \times U(i)^{\lambda_U(t)} \times D(i)^{\lambda_D} & t \geq 1 \\ U(i)^{\lambda_U(0)} & t = 0 \end{cases} \quad (\text{Eq. 4})$$

As described in Section 4.1.2, during retrieval, this step is significantly generalized to provide an extremely powerful, general, and efficient mechanism for dealing with arbitrary, nonlinear invariances, most notably, nonlinear time-warping of sequences.

4.2.1.e. Step 5: Compute the Number of Competing Hypotheses that will be Active in the Mac once the Final Code for this Frame is Activated

To motivate the need for keeping track of the number of competing hypotheses active in a mac, we consider the case of *complex* sequences, in which the same input item occurs multiple times and in multiple contexts. Figure 7 portrays a minimal example in which item B occurs as the middle state of sequences [ABC] and [DBE]. Here, the model’s single internal level, L1, consists of just one mac, with $Q=4$ CMs, each with $K=4$ cell. Figure 7a shows notional codes (SDCs) chosen on the three time steps of [ABC]. The code name convention here is that ϕ denotes a code, the superscript “1” indicates the model level at which code resides. The subscript indicates the specific moment of the sequence that the code represents; thus, it is necessary for the subscript to specify the full temporal context, from start of sequence, leading up to the current input item. Successively active codes are chained together, resulting in spatiotemporal memory traces that represent sequences. Green lines indicate the H-wts that are increased from one code to the next. Black lines indicate the U-wts that are increased from currently active pixels to currently active L1 cells (red). Thus, individual cells learn spatiotemporal inputs in correlated fashion, as whole SDCs. Learning is described more thoroughly in Section 0.

As portrayed in Figure 7b, if [ABC] has been previously learned, then when item B of another sequence, [DBC], is encountered, the CSA will generally cause a different SDC, here, ϕ_{DB}^1 , to be chosen. ϕ_{DB}^1 will be H-associated with whatever code is activated for the next item, in this case ϕ_{DBE}^1 for item E. This choosing of codes in a context-dependent way (where the dependency has no fixed Markov order and in practice can be extremely long), enables subsequent recognition of complex sequences without confusion.

However, what if in some future recognition test instance, we prompt the network with item B, i.e., as the first item of the sequence, as shown in Figure 7c? In this case, there are no active H-wts and so the computation of local support (Eq. 4) depends only on the U-wts. But, the pixels comprising item B have been fully associated with the two codes, ϕ_{AB}^1 and ϕ_{DB}^1 , which have been assigned to the two moments when item B was presented, [AB] and [DB]. We show the two maximally implicated (more specifically, maximally U-implicated) cells in each CM as orange to indicate that a choice between them in each CM has not yet been made. However, by the time the CSA completes for the frame when item B is presented, one winner must be chosen in each CM (as will become clear as we continue to explain the CSA throughout the remainder of Section 3.2). And, because it is the case in each CM, that both orange cells are equally implicated, we choose winners randomly between them, resulting in a code that is an equal mix

of the winners from ϕ_{AB}^1 and ϕ_{DB}^1 . In this case, we refer to the mac as having multiple competing hypotheses active (MCHs), where we specifically mean that all the active hypotheses (in this case, just two) are approximately equally strongly active.

The problem can now be seen at the right of Figure 7c when C is presented. Clearly, once C is presented, the model has enough information to know which of the two learned sequences, or more specifically, which particular moment is intended, [ABC] rather than [DBE]. However, the cells comprising the code representing that learned moment, ϕ_{ABC}^1 , will, at the current test moment (lower inset in Figure 7c), have only half the active H-inputs that they had during the original learning instance (i.e., upper inset in Figure 7c). This leads, once processed through steps 2b, 3b, and 4, to V values that will be far below $V=1$, for simplicity, let's say $V=0.5$, for the cells comprising ϕ_{ABC}^1 . As will be explained in the remaining CSA steps, this ultimately leads to the model *not* recognizing the current test trial moment [BC] as equivalent to the learning trial moment [ABC], and consequently, to activation of a new code that could in general be arbitrarily different from ϕ_{ABC}^1 .

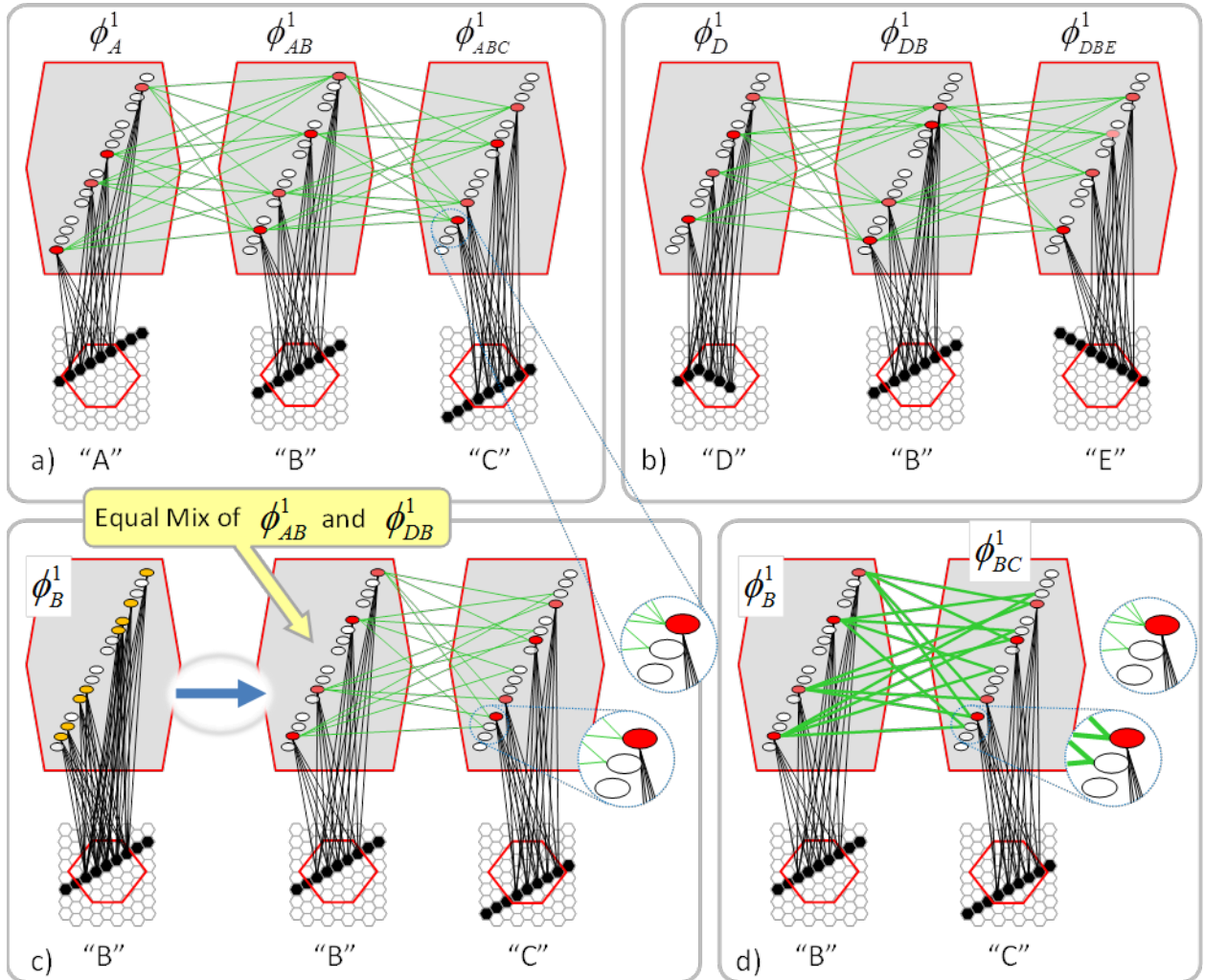


Figure 7: Illustration of Method for Handling MCHs

However, there is a fairly general solution to this problem where multiple competing hypotheses are present in an active mac code, e.g., in the code for B indicated by the yellow call-out. The mac can easily detect when an MCH condition exists. Specifically, it can tally the number cells with $V=1$ —or, allowing some slight tolerance for considering a cell to be *maximally* implicated, cells with $V(i) > V_\zeta$, where V_ζ is close to 1, e.g., $V_\zeta = 0.95$ —in each of its Q CMs, as in Eq. 5a. It can then sum ζ_q over all Q CMs and divide by Q (and round to the nearest integer, “**rni**”), resulting in the number of MCHs active in the mac, ζ , as in Eq. 5b. In this example, $\zeta = 2$, and the principle by which the H-input conditions, specifically the h-summations, for the cells in ϕ_{ABC}^1 on this test trial moment [BC] can be made the same as they were during the learning trial moment [ABC], is simply to multiply all outgoing H-signals from ϕ_B^1 by $\zeta = 2$. We indicate the *inflated* H-signals by the thicker green lines in the lower inset at right of Figure 7d. This ultimately leads to $V=1$ for all four cells comprising ϕ_{ABC}^1 and, via the remaining steps of the CSA, reinstatement of ϕ_{ABC}^1 with very high probability (or with certainty, in the simple retrieval mode described in Section 4.1.3), i.e., with recognition of test trial moment [BC] as equivalent to learning trial moment [ABC]. The model has successfully gotten through an ambiguous moment based on presentation of further, disambiguating inputs.

We note here that uniformly boosting the efferent H-signals from ϕ_B^1 also causes the h-summations for the four cells comprising the code ϕ_{DBE}^1 to be the same as they were in the learning trial moment [DBE]. However, by Eq. 4, the V values depend on the U-inputs as well. In this case, the four cells of ϕ_{DBE}^1 have u-summations of zero, which leads to $V=0$, and ultimately to essentially zero probability of any of these cells winning the competitions in their respective CMs. Though we don’t show the example here, if on the test trial, we present E instead of C after B, the situation is reversed; the u-summations of cells comprising the code ϕ_{DBE}^1 are the same as they were in the learning trial moment [DBE] whereas those of the cells comprising the code ϕ_{ABC}^1 are zero, resulting with high probability (or certainty) in reinstatement of ϕ_{DBE}^1 .

$$\zeta_q = \sum_{i=0}^K \left[V(i) > V_\zeta \right] \quad (\text{Eq. 5a})$$

$$\zeta = \text{rni} \left(\sum_{j=0}^{Q-1} \zeta_q / Q \right) \quad (\text{Eq. 5b})$$

4.2.1.f. Step 6: Compute Correction Factor for MCHs to be Applied to Efferent Signals from this Mac

The example in Figure 7 was rather clean in that it involved only two sequences having been learned, containing a total of six moments, [A], [AB], [ABC], [D], [DB], and [DBE], and very little pixel-wise overlap between the items. Thus, cross-talk between the stored codes was minimized. However, in general, macs will store far more codes. If for example, the mac of Figure 7 was asked to store 10 moments where B was presented, then, if we prompted the network with B as the first sequence item, we would expect almost all cells in all CMs to have

$V=1$. As discussed in Step 2, when the number of MCHs (ζ) in a mac gets too high, i.e., when the mac is *muddled*, its efferent signals will generally only serve to decrease signal-to-noise ratio (SNR) in target macs (including itself on the next time step via the recurrent H-wts) and so we disregard them. Specifically, when ζ is small, e.g., two or three, we want to boost the value of the signals coming from all active cells in that mac by multiplying by ζ (as in Figure 7d).

However, as ζ grows beyond that range, the expected overlap between the competing codes increases and to approximately account for that, we begin to diminish the boost factor as in Eq. 6, where A is an exponent less than 1, e.g., 0.7. Further, once ζ reaches a threshold, B , typically set to 3 or 4, we multiply the outgoing weights by 0, thus effectively disregarding the mac completely in downstream computations. We denote the correction factor for MCHs as $F(\zeta)$, defined as in Eq. 6. We also use the notation $F(\zeta(j, t))$ as in Eq. 2, where $\zeta(j, t)$ is the number of hypotheses tied for maximal activation strength in the owning mac of a pre-synaptic cell, j , at time (frame) t .

$$F(\zeta) = \begin{cases} \zeta^A & 1 \leq \zeta \leq B \\ 0 & \zeta > B \end{cases} \quad (\text{Eq. 6})$$

4.2.1.g. Step 7: Determine the Maximum Local Support in each of the Mac's CMs

Operationally, this step is quite simple: simply find the cell with the highest V value, \hat{V}_j , in each CM, C_j , as in Eq. 7. Multiple cells in a CM may be tied for \hat{V}_j .

$$\hat{V}_j = \max_{i \in C_j} \{V(i)\} \quad (\text{Eq. 7})$$

Conceptually, the cell with \hat{V}_j in a CM is the cell most implicated by the mac's total input (multiple cells can be tied for \hat{V}_j), or in other words, the *most likely* winner in the CM. In fact, in the simple retrieval mode (Section 4.1.3), the cell with \hat{V}_j in each CM is chosen winner.

4.2.1.h. Step 8: Compute the Familiarity of the Mac's Overall Input

The average, G , of the maximum V 's across the mac's Q CMs is computed as in Eq. 8: G is a measure of the *familiarity* of the macs overall input. This is done on every time step (frame), so we sometimes denote G as a function of time, $G(t)$. And, G is computed independently for each activated mac, so we may also use more general notation that indicates mac as well.

$$G = \sum_{q=1}^Q \hat{V}_k / Q \quad (\text{Eq. 8})$$

The main intuition motivating the definition and use of G is as follows. If the mac's current input moment has been experienced in the past, then all active afferent weights (U , H , and D) to the code activated in that instance would have been increased. Thus, in the current moment, all Q cells comprising that code will have $V=1$. Thus, $G=1$. Thus, a *familiar* moment must always result in $G=1$ (assuming that MCHs are accounted for as described above). On the other hand, suppose that the current *overall* input moment is novel, even if sub-components of the current overall input have been experienced exactly before. In this case, provided that few enough codes have been stored in the mac (so that crosstalk remains sufficiently small), there will be at least some CMs, C_j , for which \hat{V}_j is significantly less than 1. Thus, $G < 1$. Moreover, as the examples in the Results section will show, G correlates with the familiarity of the overall mac input. Thus, G measures the familiarity, or inverse novelty, of the global input to the mac.

Note that in the brain, this step requires that the Q cells with $V = \hat{V}_j$ become active (i.e., spike) so that their outputs can be summed and averaged. This constitutes the first of two rounds of competition that occurs within the mac's CMs on each execution of the CSA. However, as explained herein, this set of Q cells will, in general, *not* be identical to (and can often be substantially different from, especially when $G \approx 0$) the finally chosen code for this execution of the CSA (i.e., the code chosen in Step 12).

4.2.1.i. Step 9: Determine the Expansivity/Compressivity of the I/O Function to be used for the Second and Final Round of Competition within the Mac's CMs

Determine the range, η , of the sigmoid activation function, which transforms a cell's V value into its relative (within its own CM) probability of winning, ψ . We refer to that transform as the V -to- ψ map. We refer to χ as the *sigmoid expansion factor* and γ as the *sigmoid expansion exponent*.

$$\eta = 1 + \left(\left[\frac{G - G^-}{1 - G^-} \right]^+ \right)^\gamma \times \chi \times K \quad (\text{Eq. 9})$$

As noted several times earlier, the overall goal of the CSA when in learning mode is to assign codes to a mac's inputs in adherence with the SISC property, i.e., more similar overall inputs to a mac are mapped to more highly intersecting SDCs. Given that G represents, to first approximation, the similarity of the closest-matching stored input to the current input, we can restate the goal as follows.

1. as G goes to 1, meaning the input X is completely familiar, we want the probability of reinstating the code ϕ_x that was originally assigned to represent X , to go to 1. It is the cells comprising ϕ_x , which are causing the high G value. But these are the cells with the maximal V 's ($V = \hat{V}_j = 1$) in their respective CMs. Thus, within each CM,

C_j , we want to increase the probability of picking the cell with $V = \hat{V}_j$ relative to cells with $V < \hat{V}_j$, i.e., we want to transform the V 's via an *expansive* nonlinearity

2. as G goes to 0 (completely novel input), we want the set of winners chosen to have the minimum average intersection with all stored codes. We can achieve that by choosing the winner in each CM from the uniform distribution, i.e., by making all cells in a CM equally likely to win, i.e., transform the V 's via a maximally *compressive* nonlinearity.

The first goal is met by making the activation function a very expansive nonlinearity. Figure 8 shows how the expansivity of the V -to- ψ map affects cell win probability, and indirectly, whole-code reinstatement probability. All nine panels concern a small example mac with $Q=6$ CMs each comprised of $K=7$ cells. The panels show hypothetical V and ρ vectors over the cells of the CMs, across two parametrically varying conditions: model “age” (across columns), which we can take as a correlate of the number of stored codes and thus, of the amount of interference (crosstalk) between codes during retrieval, and expansivity (η) (across rows) of the V -to- ψ map. As described shortly, the V values are first transformed to relative probabilities (ψ) (Step 10), which are then normalized to absolute probabilities (ρ) (Step 11). In all panels, the example V vector in each CM has one cell with $V=1$ (pink bars). Thus, by Step 8, all panels correspond to a $G=1$ condition. The other six cells (black bars) in each CM are assigned uniformly randomly chosen values in defined intervals that depend on the age of the model (i.e., amount of input experienced). The intervals for “Early”, “Middle”, and “Late”, are $[0.0, 0.1]$, $[0.1, 0.5]$, and $[0.2, 0.8]$, respectively, simulating the increasing crosstalk with age.

For each age condition, we show the effects of using a V -to- ψ map with three different η values. Note that in actual operation (specifically, Step 9), all panels would be processed with a V -to- ψ map with the maximal η value (again, because $G=1$ in all panels). But our purpose here is just to show the consequences on the final ρ distribution for a given V distribution (the V distribution is the same for all three rows in any given column) as a function of η . And, note that the minimum ψ value in all cases is 1. Thus, for the “Early” column, the highly expansive V -to- ψ map ($\eta=300$) (top row) results in a $300/306 \approx 98\%$ probability of selecting the cell with $V=1$ (pink) in each CM. This results in a $(300/306)^6 \approx 89\%$ probability of choosing the pink cell in *all* $Q=6$ CMs, i.e., of reinstating the *entire* correct code. In the second row, η is reduced to 30. Each of the six black cells ultimately ends up with a $1/36$ probability of winning and the pink cell, with a $30/36 = 5/6$ win probability. In this case the likelihood of reinstating the *entire* correct code, is $(5/6)^6 \approx 33\%$. In the bottom row, $\eta=1$, i.e., the V -to- ψ map has been collapsed to the constant function, $\psi=1$. As can be seen, all cells, including the cell with $V=1$ become equally likely to be chosen winner in their respective CMs.

Greater crosstalk can clearly be seen in the “Middle” condition. Consequently, even for $\eta=300$, several of the cells with non-maximal V end up with significant final probability ρ of being chosen winner in their respective CMs. The ρ -distributions are slightly further compressed (flatter) when $\eta=30$, and completely compressed when $\eta=1$ (bottom row). The “Late” condition is intended to model a later period of the life of the model, after many memories (codes) have been stored in this mac. Thus, when the input pattern associated with any of those stored codes

is presented again, many of the cells in each CM will have an appreciable V value (again, here they are drawn uniformly from $[0.2, 0.8]$). In this condition, even if $\eta=300$, the probability of selecting the correct cell (pink) in each CMs is close to chance, as is the chance of reinstating the entire correct code. And the situation only gets worse for lower η values.

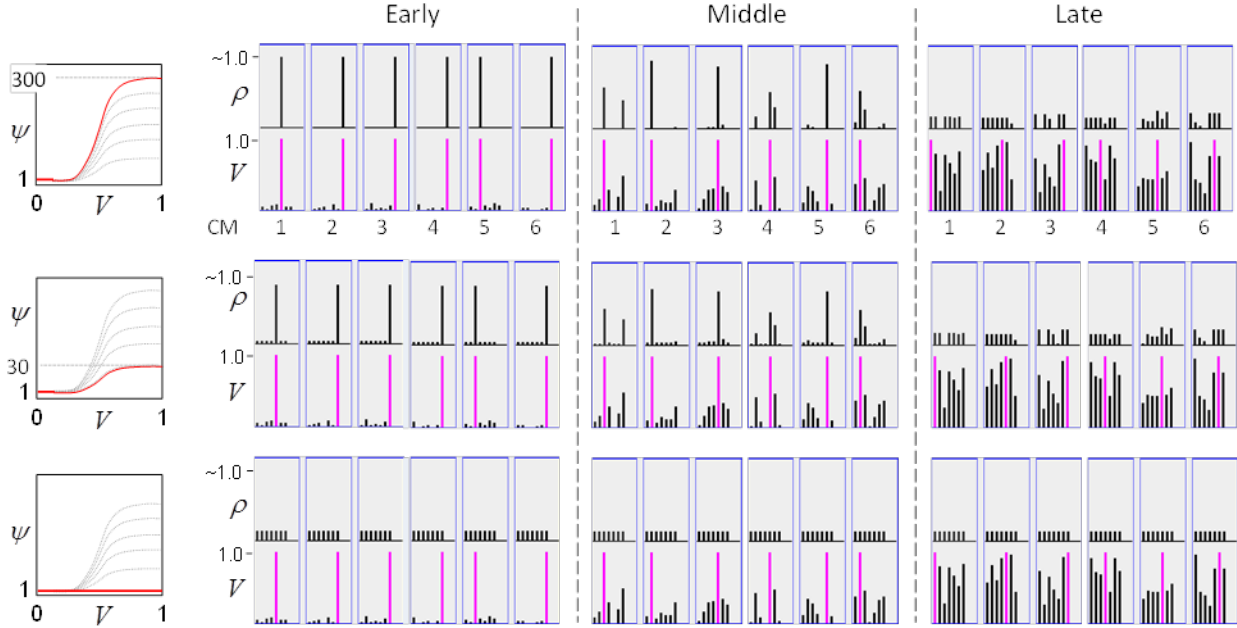


Figure 8: Consequences on Probabilistic Code Selection of Amount of Prior Learning and Varying Characteristics of G -based Sigmoid Transform

Note that for any particular V distribution in a CM, the relative increase to the final probability of being chosen winner is a smoothly and faster-than-linearly increasing (typically, $\gamma \geq 2$) function of G . Thus, in each CM, the probability that the most highly implicated (by the mac's total input) cell (those corresponding to the pink bars in Figure 8) wins increases smoothly as G goes to 1. (Strictly, this is true only for the portion of the sigmoid nonlinearity with slope > 1). The initial (left) and final (right) portions of the sigmoid are compressive ranges.) And since the overall code is just the result of the Q independent draws, it follows that the expected intersection of the code consisting of the Q most highly implicated cells, i.e., the code of the closest-matching stored input, with the finally chosen code is also an increasing function of G , i.e., thus realizing the “SISC” property.

4.2.1.j. Step 10: Apply the Modulated Activation Function to all the Mac's Cells, Resulting in a Relative Probability Distribution of Winning over the Cells of each CM

Apply sigmoid activation function to each cell. Note: the sigmoid collapses to a constant function, $\psi(i) = 1$, when $\eta = 1$ (i.e., when $G < G^-$).

$$\psi(i) = \frac{(\eta - 1)}{(1 + \sigma_1 e^{-\sigma_2(V(i) - \sigma_3)})^{\sigma_4}} + 1 \quad (\text{Eq. 10})$$

In a more general development, the CSA could include additional prior steps for setting any of the other sigmoid parameters, σ_1 , σ_2 , σ_3 , and σ_4 , all of which interact to control overall sigmoid expansivity and shape. In particular, in the current implementation, the horizontal position of the sigmoid's inflection point is moved rightward as additional codes are stored in a mac. Figure 9 shows that doing so greatly increases the probability of choosing the correct cell in each CM and thus, of reinstating the entire correct code, even when many codes have been stored in the mac. In the “Middle” condition, even if $\eta=30$, the probability of choosing the pink cell in each CM is very close to 1. For the “Late” condition, setting $\eta=30$ significantly improves the situation relative to the top right panel of Figure 8 and setting $\eta=300$ makes the probability of choosing the correct cell close to 1 in four of the six CMs. Thus, we have a mechanism for keeping memories accessible for longer lifetimes.

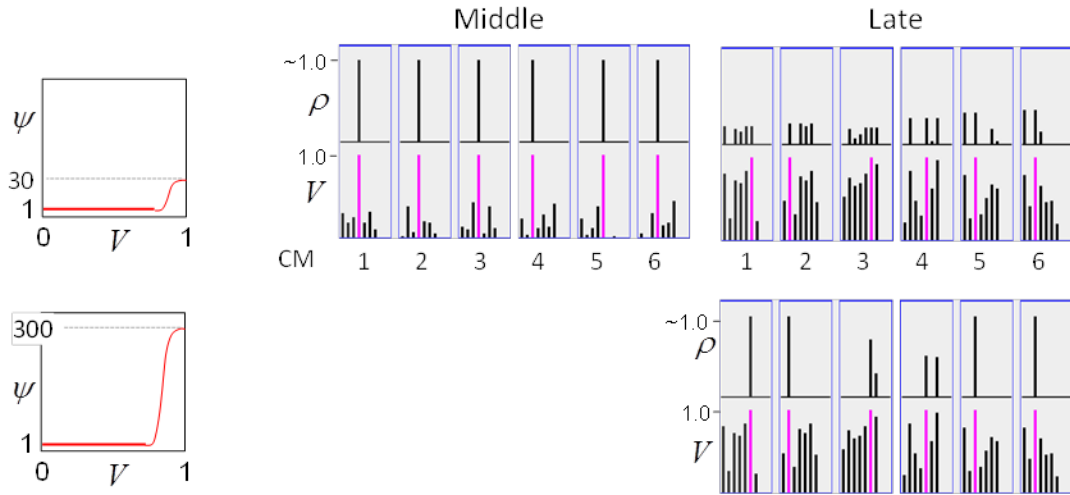


Figure 9: More Rightward Sigmoid Inflection Point Protects Against Mounting Crosstalk

4.2.1.k. Step 11: Convert Relative Win Probability Distributions to Absolute Distributions

In each of the mac's CMs, the ψ values of the cells are converted to true probabilities of winning (ρ) and the winner is selected by drawing from the ρ distribution, resulting in a final SDC, ϕ , for the mac, as in Eq. 11.

$$\rho(i) = \frac{\psi(i)}{\sum_{k \in CM} \psi(k)} \quad (\text{Eq. 11})$$

4.2.1.l. Step 12: Pick Winners in the Mac's CMs, i.e., Activate the SDC

The last step of the CSA is just selecting a final winner in each CM according to the ρ distribution in that CM, i.e., soft max. This is the second round of competition. Our hypothesis that the canonical cortical computation involves two rounds of competition is a strong and falsifiable prediction of the model with respect to actual neural dynamics, which we would like to explore further.

The CSA is given in Table 2.

Table 2. The CSA during Learning

	Equation	Short Description
1	$Active(m) = \begin{cases} true & \Upsilon(m) < \delta(m) \\ true & \pi_U^- \leq \pi_U(m) \leq \pi_U^+ \\ false & otherwise \end{cases}$	Determine if mac m will become active.
2	$u(i) = \sum_{j \in RF_U} x(j, t) \times F(\zeta(j, t)) \times w(j, i)$ $h(i) = \sum_{j \in RF_H} x(j, t-1) \times F(\zeta(j, t-1)) \times w(j, i)$ $d(i) = \sum_{j \in RF_D} x(j, t-1) \times F(\zeta(j, t-1)) \times w(j, i)$	Compute the raw U, H, and D input summations.
3	$U(i) = \begin{cases} \max(1, u(i)/\pi_U^- \times w_{\max}) & L = 1 \\ \max(1, u(i)/\min(\pi_U^-, \pi_U^*) \times Q \times w_{\max}) & L > 1 \end{cases}$ $H(i) = \max(1, h(i)/\min(\pi_H^-, \pi_H^*) \times Q \times w_{\max})$ $D(i) = \max(1, d(i)/\min(\pi_D^-, \pi_D^*) \times Q \times w_{\max})$	Compute normalized, filtered input summations.
4	$V(i) = \begin{cases} H(i)^{\lambda_H} \times U(i)^{\lambda_U^{(t)}} \times D(i)^{\lambda_D} & t \geq 1 \\ U(i)^{\lambda_U^{(0)}} & t = 0 \end{cases}$	Compute local evidential support for each cell.
5a	$\zeta_q = \sum_{i=0}^K [V(i) > V_\zeta]$	(a) Compute #cells representing a maximally competing hypothesis in each CM. (b) Compute # of maximally active hypotheses, ζ , in the mac.
5b	$\zeta = \sum_{j=0}^{Q-1} \zeta_q / Q$	
6	$F(\zeta) = \begin{cases} \zeta^A & 1 \leq \zeta \leq B \\ 0 & \zeta > B \end{cases}$	Compute the <i>multiple competing hypotheses</i> (MCH) correction factor, $F(\zeta)$, for the mac.
7	$\hat{V}_j = \max_{i \in C_j} \{V(i)\}$	Find the max V , \hat{V}_j , in each CM, C_j .
8	$G = \sum_{q=1}^Q \hat{V}_k / Q$	Compute G as the average \hat{V} value over the Q CMs.
9	$\eta = 1 + \left(\left[\frac{G - G^-}{1 - G^-} \right]^+ \right)^{\gamma} \times \chi \times K$	Determine the expansivity of the sigmoid activation function.
10	$\psi(i) = \frac{(\eta - 1)}{(1 + \sigma_1 e^{-\sigma_2 (V(i) - \sigma_3)})^{\sigma_4}} + 1$	Apply sigmoid activation function (which collapses to the constant function when $G < G^-$) to each cell.

	Equation	Short Description
11	$\rho(i) = \frac{\psi(i)}{\sum_{k \in CM} \psi(k)}$	In each CM, normalize the relative probabilities of winning (ψ) to final probabilities (ρ) of winning.
12	Select a final winner in each CM according to the ρ distribution in that CM, i.e., soft max.	

4.2.1.m. Learning Policy and Mechanics

Broadly, Sparsey’s learning policy can be described as Hebbian with passive weight decay. As noted earlier, the model’s synapses are *effectively binary*. By this we mean that although the weight range is $[0,127]$, the several learning related properties conspire to cause the asymptotic weight distribution to tend towards having two spikes, one at 0 and the other at $w_{\max} = 127$, thus effectively being binary.

In actuality, a synapse’s weight, $w(j,i)$, where j and i index the pre- and post-synaptic cells, respectively, is determined by two primary variables, its *age*, $\sigma(j,i)$, which is the number of time steps (e.g., video frames) since it was last increased, and its *permanence*, $\theta(j,i)$, which measures how resistant to decrease the weight is (i.e., the passive decay rate). Figure 10 provides a visual depiction of the learning law; weight age is indexed by column and permanence by row. The learning law is implemented as follows. Whenever a synapse’s pre- and postsynaptic cells are coactive [i.e., a “pre-post correlation”, $a(j) = 1 \wedge a(i) = 1$], its age is set to zero, as in Eq. 12a., which has the effect of setting its weight to w_{\max} . This can be seen in the “weight table” of Figure 10 in that an age of zero always maps to w_{\max} . Otherwise, $\sigma(j,i)$ increases by one on each successive time step (across all frames of all sequences presented) on which there is no pre-post correlation (Eq. 12c), stopping when it gets to the maximum age, σ_{\max} (Eq. 12d). Also note that once a synapse has reached maximum permanence, θ_{\max} , its age stays at zero, i.e., its weight stays at w_{\max} (Eq. 12b). At any point, the synapse’s weight, $w(j,i)$, is gotten by dereferencing $\sigma(j,i)$ and $\theta(j,i)$ from the weight table shown in Figure 10.

The intent of the decay schedule (for any permanence value) is to keep the weight at or near w_{\max} for some initial window of time (number of time steps), $T_{\sigma}(\theta)$, and then allow it to decay increasingly rapidly toward zero. Thus, the model “assumes” that a pre-post correlation reflects an important / meaningful event in the input space and therefore strongly embeds it in memory (consistent with the notion of episodic memory). If the synapse experiences a second pre-post correlations within the window $T_{\sigma}(\theta)$, its permanence is incremented as in Eq. 13 and $\sigma(j,i)$ is set back to 0 (i.e., its weight is set back to w_{\max}); otherwise the age, $\sigma(j,i)$, increases by one with each time step and the weight decreases according to the decay schedule in effect. Thus, pre-post correlations due to noise or spurious events, which will have a much longer expected time to recurrence, will tend to fade from memory. Sparsey’s permanence property is closely

related to the notion of synaptic tagging (Frey and Morris 1997, Morris and Frey 1999, Sajikumar and Frey 2004, Moncada and Viola 2007, Barrett, Billings et al. 2009)).

$$\sigma(j,i) = \begin{cases} 0 & , \quad a(j) = 1 \wedge a(i) = 1 \\ 0 & , \quad \theta(j,i) = \theta_{\max} \\ \sigma(j,i) + 1 & , \quad a(j) = 0 \vee a(i) = 0 \\ \sigma(j,i) & , \quad \sigma(j,i) = \sigma_{\max} \end{cases} \quad \begin{array}{l} \text{(Eq. 12a)} \\ \text{(Eq. 12b)} \\ \text{(Eq. 12c)} \\ \text{(Eq. 12d)} \end{array}$$

$$\theta(j,i) = \begin{cases} \theta(j,i) + 1 & , \quad a(j) = 1 \wedge a(i) = 1 \wedge \sigma(j,i) \leq T_{\sigma}(\theta(j,i)) \\ \theta(j,i) & , \quad \text{otherwise} \end{cases} \quad \text{(Eq. 13)}$$

The exact parametric details are less important, but as can be seen in the weight table, the decay rate decreases with $\theta(j,i)$ and the window, $T_{\sigma}(\theta)$, within which a second pre-post correlation will cause an increase in permanence, increases with $\theta(j,i)$ (three example value shown). Permanence can only increase and in our investigations thus far, we typically make a synaptic weight completely permanent on the second or third within-window pre-post correlation [$\theta_{\max} = 1$ or $\theta_{\max} = 2$, respectively]. The justification of this policy derives from two facts: a) a mac's input is a sizable set of co-active cells; and b) due to the SISC property, the probability that a weight will be increased correlates with the strength of the statistical regularity of the input (i.e., the structural permanence of the input feature) causing that increase. These two facts conspire to make the expected time of recurrence of a pre-post correlation exponentially longer for spurious / noisy events than for meaningful (i.e., due to structural regularities of the environment) events.

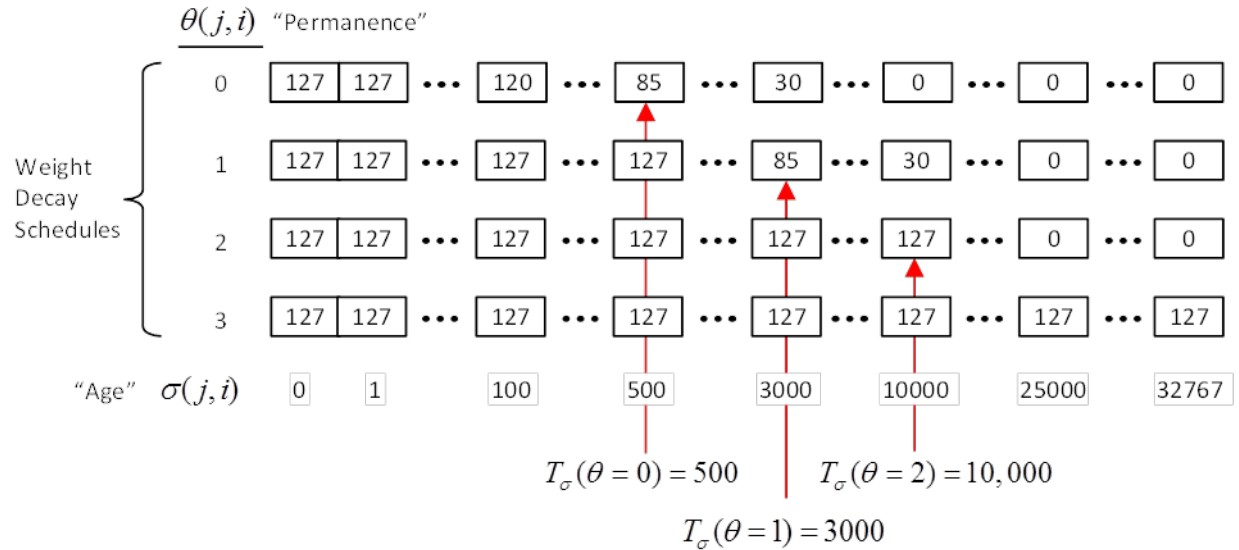


Figure 10: The "Weight Table" Indexed by Age and Permanence

If we run the model indefinitely, then eventually every synapse will experience two successive pre-post correlations occurring within any predefined window, T_σ . Thus, without some additional mechanism in place, eventually *all* afferent synapses into a mac will be permanently increased to $w_{\max} = 127$ at which point (total saturation of the afferent weight matrices) all information will be lost from the afferent matrices. Therefore, Sparsey implements a “critical period” concept, in which *all* weights leading to a mac are “frozen” (no further learning) once the fraction of weights that have been increased in any *one* of its afferent matrices crosses a threshold. This may seem a rather drastic solution to the classic trade-off that Grossberg termed the “stability-plasticity dilemma” (Grossberg 1980). However, note that: a) ‘critical periods’ have been demonstrated in the real brain in vision and other modalities (Wiesel and Hubel 1963, Barkat, Polley et al. 2011, Pandipati and Schoppa 2012); b) model parameter settings can readily be found such that in general, all synaptic matrices afferent to a mac approach their respective saturation thresholds roughly at the same time (so that the above rule for freezing a mac does not result in significantly underutilized synaptic matrices); and c) in Sparsey, freezing of learning is applied on a mac-by-mac basis. We anticipate that in actual operation, the statistics of natural visual input domains (filtered as described earlier, i.e., to binary 1-pixel wide edges) in conjunction with model principles/parameters will result in the tendency for the lowest level macs to freeze earliest, and progressively higher macs to freeze progressively later, i.e., a “progressive critical periods” concept. Though clearly, if the model as a whole is to be able to learn new inputs throughout its entire “life”, parameters must be set so that some macs, logically those at the highest levels, never freeze. We are still in the earliest stages of exploring the vast space of model parameters that influence the pattern of freezing across levels.

The ultimate test of whether the use critical periods as described above is too drastic or not is how well a model can continue to perform recognition/retrieval (or perform the specific recognition/retrieval-contingent tasks with which it is charged) over its operational lifetime (which will in general entail large numbers of novel inputs), in particular, after many of its lower levels have been frozen.

4.2.2. CSA: Retrieval Mode

In this section, we will first motivate the need for introducing some complexity to the computation of G when in retrieval mode and then describe the modification. We begin by thinking about how the model should respond to test trials involving previously learned sequences corrupted in particular ways. For example, if the model has learned the sequence $S1=[\text{BOUNDARY}]$ in the past and is now presented with $S2=[\text{BOUNDRY}]$, should it decide that $S2$ is functionally equivalent to $S1$? That is, should it respond equivalently to $S2$ and $S1$? More precisely, should its internal state at the end of processing $S2$ be the same as it was at the end of processing $S1$? The reader will probably agree that it should. We all encounter spelling errors like this all the time and read right through them. Similarly, if one encountered $S3=[\text{BBOUNDARY}]$, $S4=[\text{BBOOUUNNDDAARRYY}]$, $S5=[\text{BOUNNNNNNDARY}]$, or any of numerous other variations, he/she would likely decide it was an instance of $S1$. We could think of all these variations (corruptions) simply as omissions/repetitions. However, we prefer to think of this class of corruptions as instances of the class of *nonlinearly time-warped* instances of (discrete) sequences. Thus, $S2$ can be thought of as an instance of $S1$ that is presented at the same speed as during learning up until item “D” is reached, at which time the process presenting

the items momentarily speeds up (e.g., doubles its speed) so that “A” is presented but then replaced by “R” before the model’s next sampling period. Then the process slows back down to its original speed and item “Y” is sampled. Thus S2 is a nonlinearly time-warped instance of S1. We can construct similar explanations, involving the underlying process producing the sequences undergoing a schedule of speedups and slowdowns relative to the original learning speed, for S3, S4, etc. In fact, S4 is even simpler; it’s just a uniform slowing down, to half speed, of the whole process.

Of course, there are limits to how much we want a system to generalize regarding these warpings. And the final equivalence classes, in particular for processing language, must be experience-dependent and idiosyncratic. For example, should a model think that S6=[COD] is just an instance of S7=[CLOUDS], produced twice as fast as during the learning instance? In general, probably not. Furthermore, we have not even considered in these examples the fact that the individual sequence items are actually pixel patterns which can themselves be noisy, partially occluded, etc., which would of course influence the normative category decisions. Nevertheless, the ubiquity of instances such as described above, not just in the realm of language, but of lower-level raw sensory inputs, suggests that a model have some mechanism for dealing with them, i.e., some mechanism for treating moments produced by nonlinearly time-warping as equivalent. Our explanation of the modified *G* computation in retrieval mode uses an example involving a 3-level model that has only one mac at each level. Figure 11 shows representative samples of the U, H, and D learning that occurs as the model is presented with the sequence, [BOTH]. Note that the model is unrolled in time here, i.e., the model is pictured at four successive time steps and in particular, the origin and destination cell populations of the increased H synapses (green) are the same. This figure illustrates several key concepts. First, learning a sequence involves increasing the H-wts from the previously active code to the currently active code. The D-wts (magenta) are also increased from the previously active code (in this case, in the L2 mac) to the currently active destination code in the L1 mac. Note however that the U-wts (blue) are increased from the currently active input (L0 code) to the currently active L1 code. We show the full set of afferent U, H, and D wts that are increased for one cell—the winner in the upper left CM of the L1 mac—at each time step. Thus, this figure emphasizes that, on each moment, individual cells become associated with their entire afferent input (spatiotemporal context) in one fell swoop. Though we only show this occurring for one cell on each frame, all winners in a mac code will receive the same weight increases simultaneously. Thus we can say not only that individual cells become associated with the mac’s entire spatiotemporal contexts but that whole mac codes become associated with the mac’s entire spatiotemporal contexts.

The second key concept illustrated is progressive persistence, in this case, that L2 codes persist for twice as long as L1 codes. Cell color in this figure is used to make persistence clear. Thus, the first L2 code that becomes active D-associates with two L1 codes. And, because of the modeling decision that D-wts are increased from previously active to currently active codes, the two L1 codes are those at $t=2$ and $t=3$. The second L2 code to become active (orange) D-associates with the L2 code at $t=3$ and would associate with a $t=4$ L1 code if one occurred.

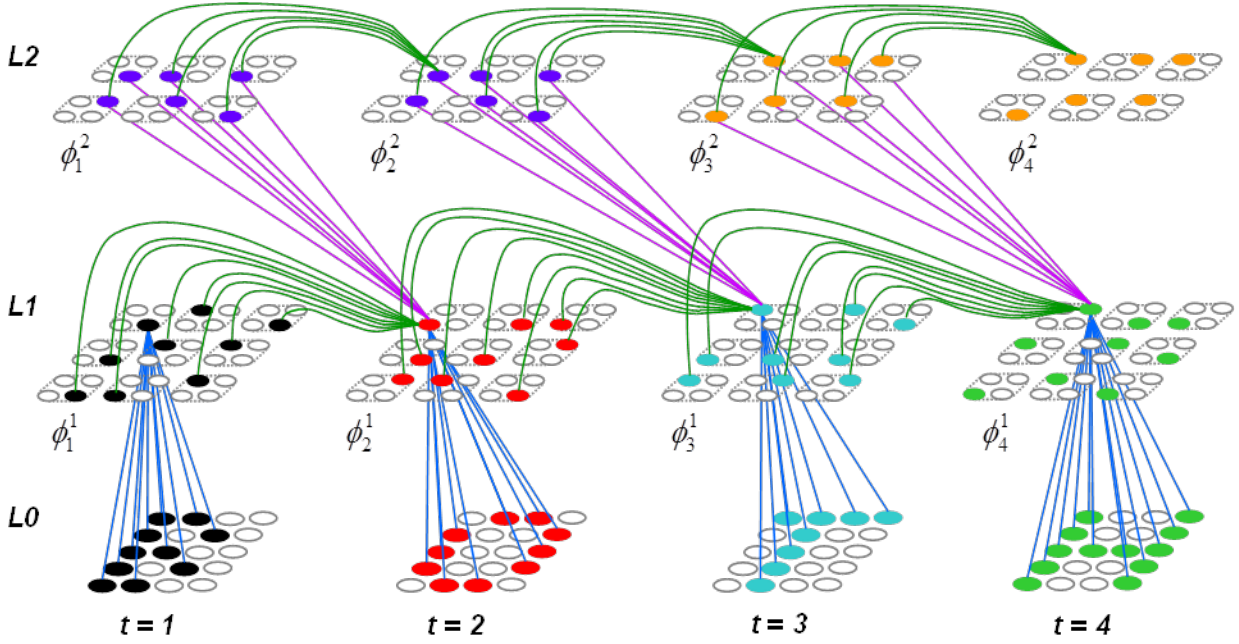


Figure 11: Formation of Hierarchical Spatiotemporal Memory Trace, Unrolled in Time

The trace of Figure 11 can be said to have been produced using both *chaining* (increasing H-wts between successively active codes at the same level) and *chunking* (increasing U and D wts between single higher-level (L2) codes and multiple lower-level (L1) codes).

Having illustrated (in Figure 11) the nature of the hierarchical spatiotemporal memory trace that the model forms for [BOTH], Figure 12 compares model conditions when processing one particular moment—the second moment—of a test trial that is identical to the learning trial (Figure 12a) to conditions when processing the second moment of a time-warped instance of the learning trial—specifically, a moment at which the item that originally appeared as the third item of the learning trial, “T”, now appears as the second item immediately after “B”, i.e., “O” has been omitted (Figure 12b). We can represent the two test trial moments as [BO] and [BT], respectively, where bolding indicates the frame currently being processed and the non-bolded letters indicate the context leading up to the current moment. The first thing to say is that the second moment of the time-warped instance is simply a novel moment. Thus, the caveat we mentioned above applies. That is, deciding whether a particular novel input moment should be considered a time-warped instance of a known moment or as a new moment altogether cannot be done absolutely.

Figure 12a shows the case where the test trial moment [BO] is identical to the learning trial moment [BO]. The main point to see here is that, given the weight increases that will have occurred on the learning trial, all three input vectors, U, H, and D, will be maximal (equal to 1) for the red cell (which is in ϕ_2^1) in each L1 CM. At right (yellow), we zoom in on the conditions only for the upper left L1 CM, but the conditions are statistically similar for all L1 CMs. We show that for the red cell, $U=1$, $H=1$, and $D=1$. The blue cell (which is in ϕ_3^1) also has maximal D-support and the blue, green, and black cells have non-zero U inputs (their U-inputs are not shown in the main figure to minimize clutter), due to the pixel overlap amongst the four input

patterns, but they all have $H=0$. Thus, according to Eq. 4 of the CSA (Table 2), the red cell has $V=U \times H \times D=1$, whereas the others have $V=0$. We refer to red cell as having a “3-way match” in that all three evidence vectors are maximal and agree. Also, we refer to the G version computed using all three input vectors as G_{HUD} . Thus, in this case, where the test moment is identical to a learned moment, CSA Eq. 4 is sufficient as is.

However as shown in Figure 12b, when an item (“O”) has been omitted with respect to the learning trial, the H and D vectors to the red cell will no longer agree with its U vector. Various policies could be imagined for handling this situation. The model could simply consider such a case as being a novel moment, [BT]. This would require no modification to the CSA. Or, as discussed earlier, the model could *check* to see whether the current moment could have resulted from a nonlinear time-warping process, and should therefore be judged identical to some previously learned moment. In this case, the current moment [BT] is identical to the learning trial moment [BOT] if we assume that the process presenting the sequence to the model sped up by 2x at $t=1$, causing the “O” to be missed.

So, how does the model *check* this possibility? It is quite simple. All it needs to do is disregard the H signals when computing the V 's (CSA Step 4). In other words, it “backs off” from the more stringent 3-way G_{HUD} match criterion to the more permissive 2-way G_{UD} criterion. Note that the model begins by computing the highest-order G available at the current moment, in this case, using all three input vectors. Only if that highest-order G falls below a threshold, which we typically set rather high, e.g., $G_{HUD}^+ = 0.9$, does it bother to compute the next lower order version(s) of G , i.e., G_{UD} , G_{HU} , and G_{HD} . Similarly, only if whichever 2-way version has been considered falls below another threshold, which is typically set even higher than the first, e.g., $G_{UD}^+ = 0.95$, does the model *back-off* to the next lower order match criterion.

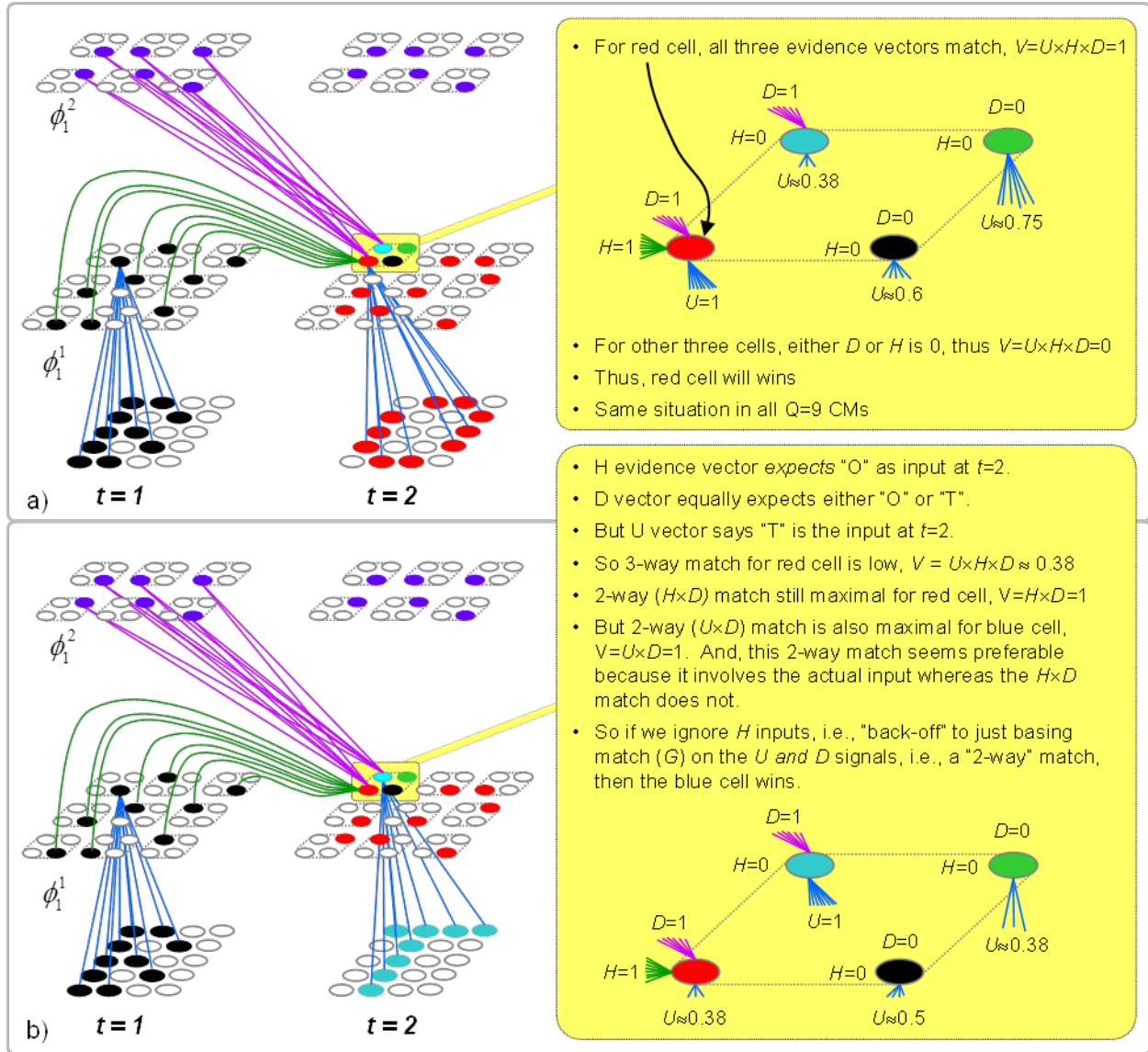


Figure 12: Motivation for the Back-Off Strategy for Computing G in Retrieval Mode

In this example, $G_{UD} = 1$, meaning that there is a code stored in the L1 mac—specifically, the set of blue cells assigned as the L1 code at $t=3$ of the learning trial (Figure 11)—which yields a perfect 2-way match. Thus, there is no need to back-off to the “1-way” match criterion, G_U . However, there are many naturally occurring instances in which backing all the way off to the lowest-order criterion (i.e., basing the V values and thus, the G , on only the U signals, ignoring the H and D signals) is appropriate. There are myriad policy considerations regarding possible precedence orders of the different G versions and whether or not and under what conditions the various versions should be considered. We are actively exploring these issues, but cannot delve into this topic in this paper.

Figure 13 completes this example by showing that the back-off policy allows the model to keep pace with nonlinearly time-warped instances of previously learned sequences. That is, the model's internal state (i.e., the codes active in the macs) can either advance more quickly (as in this example) or slow down (not demonstrated herein) to stay in sync with the sequence being presented. Figure 13a is given for comparison, showing the full memory trace that becomes active during a retrieval trial for an exact duplicate of the training trial, [BOTH]. In this case, no back-off would be required because all signals at all times would be the same during retrieval as they were during learning. Figure 13b shows the trace that obtains, using the back-off protocol, throughout presentation of the nonlinearly time-warped instance of the training trial, [BTH].

The back-off from G_{HUD} to G_{UD} occurs in the L1 mac at $t=2$ (as was described in Figure 12b). Since $G_{UD}=1$, the V -to- ψ map is made very expansive, resulting in activation, at $t=2$ of the test trial, of the code, ϕ_3^1 (blue cells), which was originally activated at $t=3$ in the learning trial. Thus, the back-off has allowed the model's internal state (in L1) to “catch up” to the momentarily sped up process that is producing the input sequence. Once ϕ_3^1 is activated, it sends U-signals to L2 (blue signals converging on orange cell in rose highlight box). This results in the L2 code, ϕ_3^2 (orange cells), being activated without requiring any back-off. That's because the L2 code from which H signals arrive at $t=2$, ϕ_1^2 (purple cells) increased its weights not only onto itself (at $t=2$ of the learning trial) but also onto ϕ_3^2 at $t=3$ of the learning trial. Thus, the six cells comprising ϕ_3^2 (orange) yield $G_{HU}=1$ (note that G_{HU} is the highest order G version available at L2 since there is no higher level). Consequently, a maximally expansive V -to- ψ map is used in the L2 mac, resulting in reinstatement of ϕ_3^2 . At this point— $t=2$ of the test trial—the entire internal state of the model (i.e., at L1 and L2) is identical to its state at $t=3$ of the learning trial (two central dashed boxes connected by double-headed black arrow): the model, as a whole, has “caught up” with the momentary speed up of the sequence. The remainder of the sequence proceeds the same as it did during learning, i.e., state at $t=3$ of retrieval trial equals state at $t=4$ of learning trial.

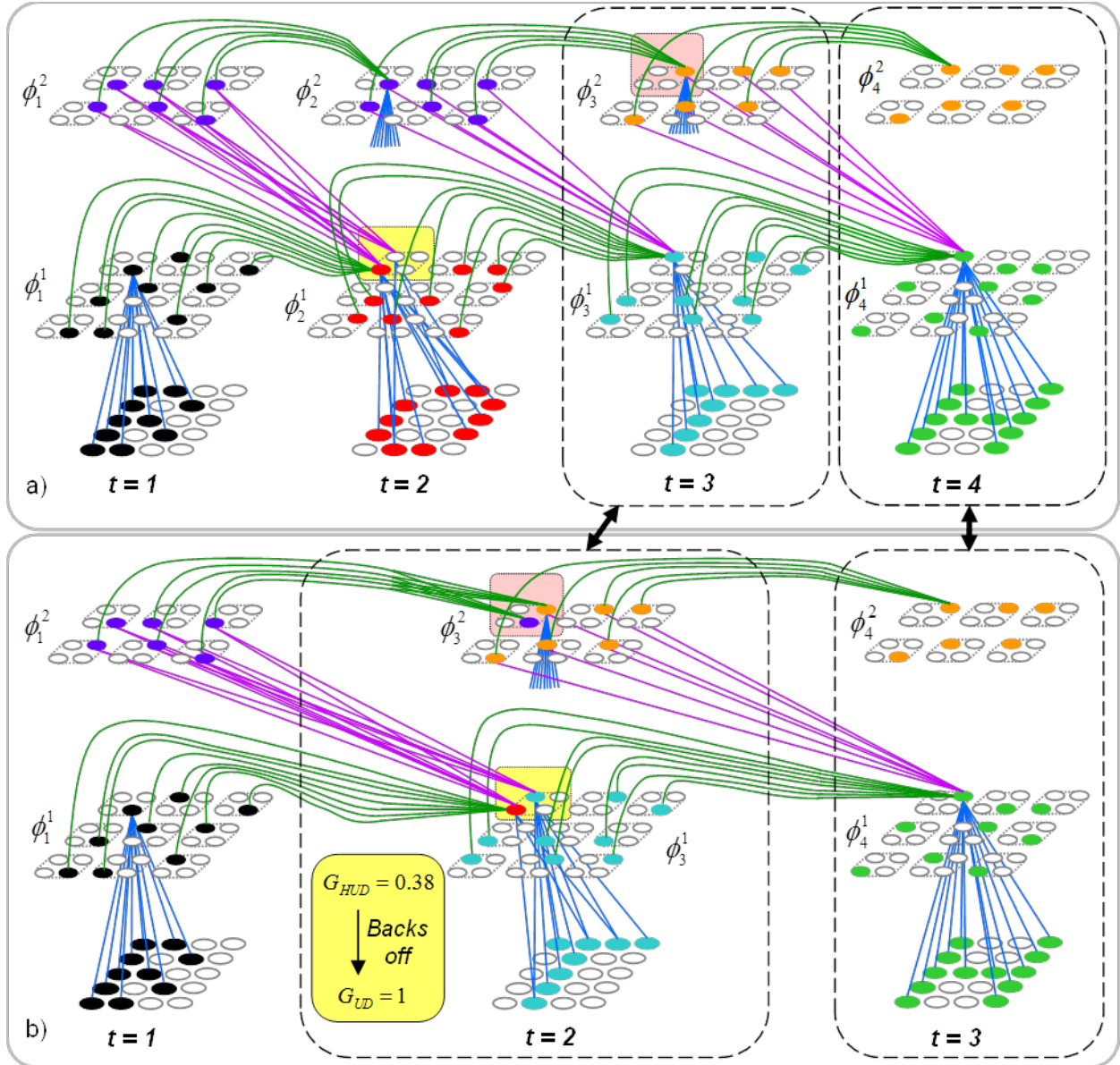


Figure 13: Back-Off Allows Internal State to Remain Synchronized with Nonlinearly Time-Warped Instances of Known Snippets

The final, and really the most important, point of this section is that Sparsey’s back-off policy *does not change* the time complexity of the CSA: it still runs with *fixed time complexity*, which is essential in terms of scalability to real-world problems. True, expanding the logic to compute multiple versions of G increases the absolute number of computer operations required by a single execution of the CSA. However, the number of possible G versions is small and more to the point, fixed. Thus, adding the back-off logic adds only a fixed number of operations to the CSA and so does not change the CSA’s time complexity.

During each execution of the CSA, *all stored codes* compete with each other. In general, the set of stored codes will correspond to moments spanning a large range of *Markov orders*. For example, in Figure 11, the four moments, [B], [BO], [BOT], and [BOTH], are stored, which are

of progressively greater Markov order. *During each moment of retrieval, they all compete.* More specifically, they all compete first using the highest-order G , and then if necessary, using progressively lower-order G 's. However, it is crucial to see that with back-off, not only are the explicitly stored (i.e., actually experienced) moments compared, but so are a far larger number of time-warped versions of the actually-experienced moments. For example in Figure 12b and Figure 13b, the moment [BT], which never actually occurred competes and wins (by virtue of back-off) over the moment [BO], which did occur. And crucially, as noted above, all these comparisons take place with fixed time complexity! Space does not permit here, but the above mechanism and reasoning generalizes to arbitrarily deep hierarchies. As the number of levels increases, with persistence doubling at each level, the space of hypothetical nonlinearly time-warped versions of actually experienced moments, which will materially compete with the actual moments (on every frame and in every mac) grows exponentially. And, we emphasize that these exponentially increasing spaces of never-actually-experienced hypotheses are *envelopes* around the actually-experienced moments: thus, the invariances implicitly represented by these envelopes are (a) learned and (b) idiosyncratic to the specific experience of the model.

4.2.3. CSA: Simple Retrieval Mode

Both the learning mode CSA and the retrieval mode CSA described above, which is just the learning mode CSA augmented by the back-off protocol, involve the G -based modification of the cell activation functions and the second, probabilistic round of competition for choosing the final code (CSA Steps 8-12, Table 2). If the model is operating as a truly autonomous agent, then it, or rather any of its constituent macs, may be presented with a truly novel input pattern at every moment experienced. Thus, a mac must be prepared to learn, i.e., assign a new SDC, at every moment.⁴ As described in earlier sections, the CSA's two competitive stages, with the second, probabilistic stage using the G -modulated cell activation functions, satisfies the requirements for autonomous operation. That is, as G decreases, the expected intersection of the final code (for the current frame) chosen with the closest matching stored code decreases to chance, which results in the occurrence of novel pre-post correlations, and thus new learning. On the other hand, as G increases towards 1, the expected intersection of the finally chosen code with the closest matching stored code increases to complete, which results in no (or at least, statistically, very few) novel pre-post correlations and thus no new learning.

However, if the model "knows" that is operating in pure retrieval mode, i.e., that at each moment each mac should simply activate the code of the learned moment that most closely matches its current input moment, then there is no advantage to having the second G -dependent probabilistic stage of competition. In fact, the optimal strategy in this case is simply to choose the cell with the highest V value in each CM. The transfer of global information (G) back into the local (within each CM) winner selection processes, which occurs in steps 8-12, does not help and in fact, can only hurt (i.e., it can only reduce the probability of the maximally likely cell in a given CM winning). Thus, in this "simple retrieval mode", in which the model knows that it will not be asked to learn anything new, the optimal algorithm is just the first seven steps of the CSA given in Table 2, but augmented with the back-off protocol described in the previous section.

⁴ Actually, in a hierarchical model faced with the prospect of possibly having to learn something new on every moment of its operational lifetime, its sufficient only that at least one mac (which would typically be at the highest level) be prepared to learn at every moment (cf. earlier discussion of critical periods).

Thus, we do not state the simple retrieval mode of the CSA separately. We will clearly indicate which of the two retrieval modes is used in the studies reported in the next section.

We emphasize that the *deterministic* “simple retrieval mode” algorithm cannot be used during learning because it would result in essentially mapping all of the mac’s input patterns to one or a very small number of codes, vastly over-utilizing only a tiny fraction of the mac’s cells and vastly decreasing the number of codes (amount of information) that can be stored in the mac.

However, based on first principles, it seems plausible that for the vast majority of Sparsey’s envisioned operational regime, i.e., the regime in which the number of codes stored in the macs (or more specifically, the fraction of synapses that have been increased) remains below a threshold, the simple retrieval mode should always do better (on average) than the probabilistic retrieval mode. Specifically, recall that in probabilistic retrieval mode, the winner in a CM is chosen *as a draw* from the V distribution. Depending on the particular shape/statistics of the V distribution, the cell with the maximum V might therefore be chosen winner only a small fraction of the time. Yet, that max- V cell is the most likely cell given the total evidence (from the U, H, and D signals) arriving at the mac. In simple retrieval mode, the max- V cell always wins. Again, provided that the fraction of the mac’s afferent synapses that have been increased remains low enough, simply choosing the max- V cell as winner yields higher expected accuracy.

5. RESULTS

5.1 Individual Macs Implement SISC

The focus of the first Technical Report delivered for this project was to show that Sparsey’s core algorithm, the CSA, maps spatiotemporal similarity in the input space into similarity in internal representation, or code, space. We refer to this property as *similar-inputs-map-to-similar-code*.. Of course, any pattern recognition system must possess this property. However, we emphasize that Sparsey, more specifically, each individual Sparsey mac, learns a particular spatiotemporal similarity measure based on its inputs. The space of possible metrics that can become embedded in a mac is constrained by various of the CSA’s parameters, but the exact metric depends on and is tuned to the statistics of the inputs actually experienced by the mac. This is in contrast to pattern recognition systems in which the similarity metric is prescribed a priori, which limits autonomy.

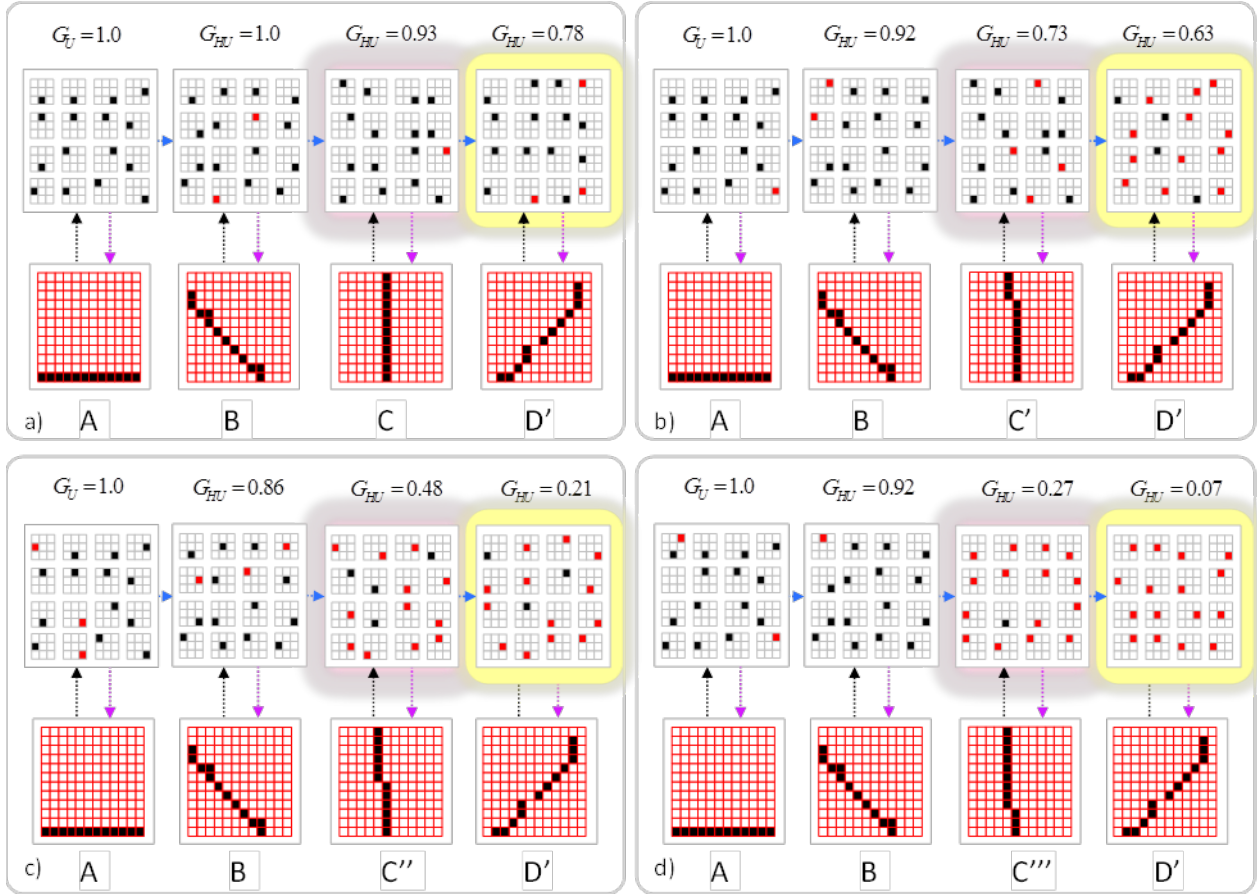


Figure 14: Demonstration of SISC Property for 4-item-long Sequences

Figure 14 shows an example of the SISC property with respect to a single mac. The spatiotemporal similarity (G_{HU}) of the final sequence moment (D') to the final moment of the learned sequence, [ABCD], falls from panel a to panel d. The size of intersection of the codes of those moments (highlighted in yellow) falls correspondingly. A similar correlation exists for the third sequence moments as well (highlighted in pink). Although mistakes are made on the first

and second moments, the number of mistakes remains statistically constant across panels for these moments. This particular relation between spatiotemporal input similarity and code similarity realized in this example depends on the specific parameters of the sigmoid mapping from cells' V values to their win probabilities (within their CMs). A more detailed analysis of the SISC property is given in Appendix B.

5.2 Simple Features Support High Class Accuracy

In our current studies, we edge filter, binarize, apply bounding box, and scale down the original (180x144, grayscale) Weizmann frames to be 42x60 pixels. We also decimate frames. The original and resulting frames for one of the Weizmann snippets is shown in Figure 15. Despite the clearly reduced information present in the inputs fed to Sparsey, relatively high classification accuracy (67%) is achieved. It is quite possible that higher classification accuracy would result given less time decimation, less spatial scaling, etc. This would be a good candidate for a follow-on research task. Appendix A provides a more detailed description of our edge filtering pipeline.

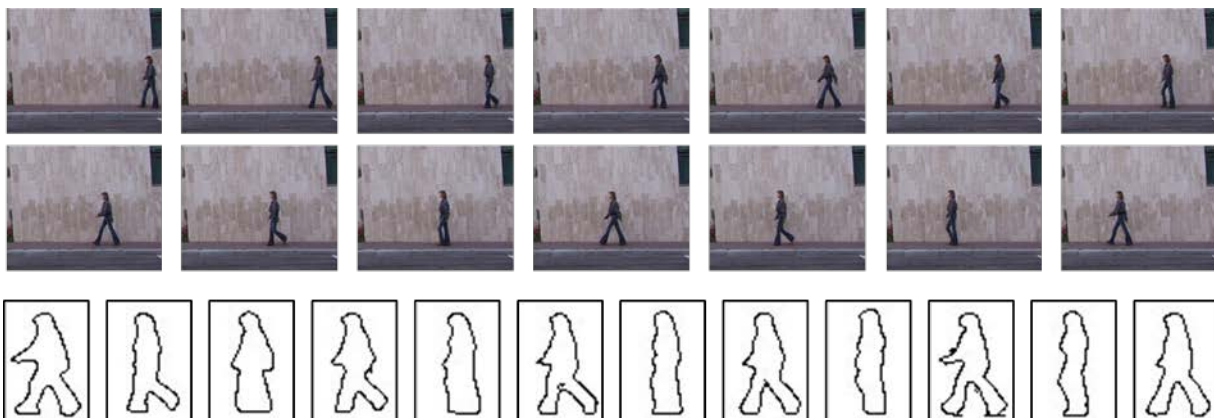


Figure 15: Original and Preprocessed Frames of a Weizmann Snippet

5.3 Sanity Tests (Test Set = Train Set)

We performed a great many “sanity test” experiments throughout the project. Such tests do more than simply show that the model can output the correct class of snippets that it has seen during training. Specifically, they show that highly detailed, hierarchical, spatiotemporal memory traces, spanning many frames, and involving thousands of precisely-timed individual cell activations can be recapitulated during testing with very high accuracy: we refer to this measure as trace accuracy, denoted by the symbol R (e.g., R^* and R^Ω), throughout. We have found that precise recapitulation of memory traces is not generally needed to support good classification in the “test \neq train” case. However, it is necessary to emulate human *episodic* memory, which is highly detailed memory for specific, generally temporally-extended, experiences. Sparsey’s (formerly TEMECOR’s) development over the past 25 years has always had the dual goals of explaining both *semantic* memory, which is essentially classification/recognition capability, and episodic memory. We describe three sanity (“train = test”) experiments here.

5.3.1. Sanity and Noisy Recognition Tests with Edge-Filtered Videos

Figure 16 shows one edge-filtered frame (84x120) from one of the 90 Weizmann snippets (left) and the corresponding 40% noisy version of the same frame. Figure 17 shows the model on a particular frame of one of the snippets, showing the U-wts (blue lines) from active pixels to the four cells (black) comprising the active code in the single active mac on this frame. The horizontal (H) weights carrying signals from the previously active code (gray cells) in a neighboring mac are shown (green arcs).

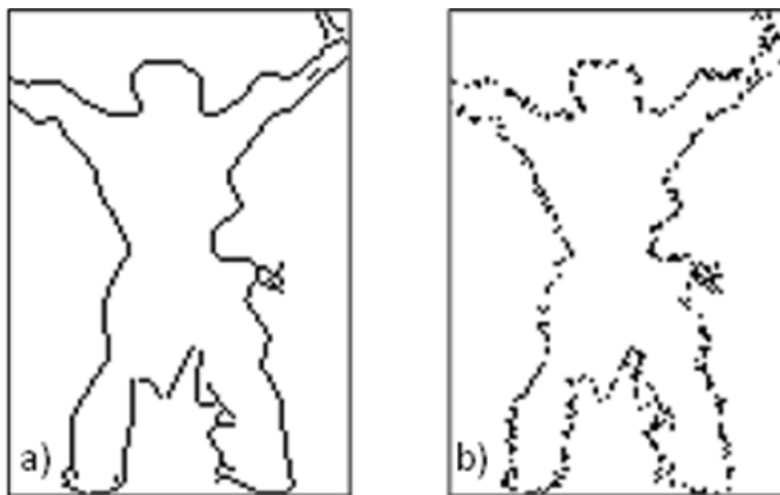


Figure 16: Original Edge-filtered Frame and Corresponding 40% Noisy Frame

The presence of only one internal level in this model forced some non-standard parameter settings, e.g., *all* L1 cells receive inputs from *all* input (L0) cells (pixels). This degree of fan-in will not be present in true hierarchical models. In any case, the model successfully learns all 90 snippets with single trials, in 187 seconds (on single hyperthread of 3.2 GHz processor, i.e., no machine parallelism), and recognizes them, in terms of both trace (93%) and classification (98%) accuracy, almost perfectly, as well as also recognizing highly noisy (40%) versions of the snippets as well (see Table 3). The training set contained 90 snippets, from 12 to 33 frames long, for a total of 1,722 frames. Web pages provide additional details of the [sanity](#) and [noise](#) experiments.

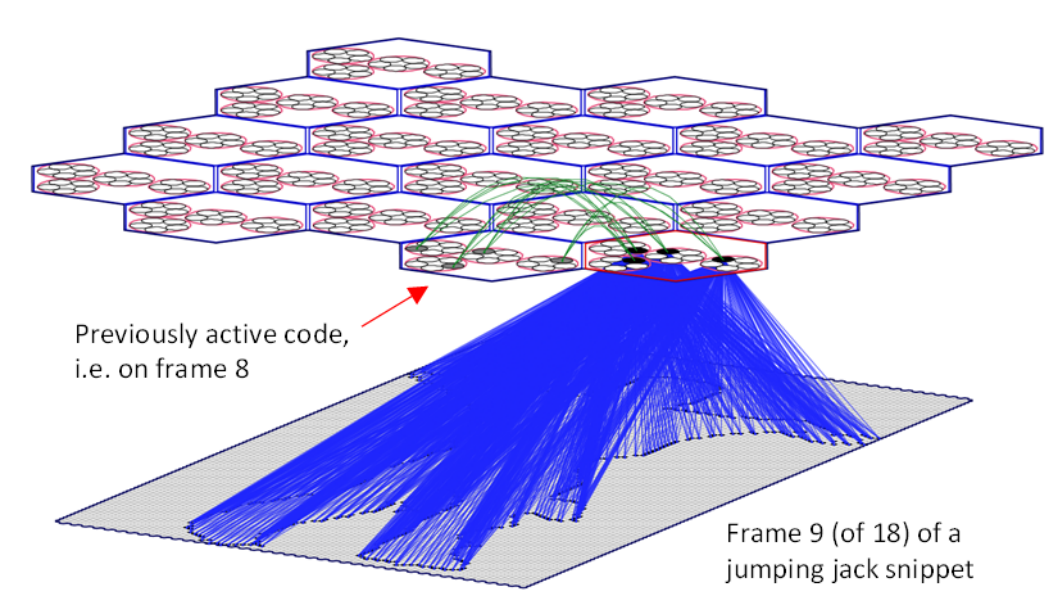


Figure 17: Snapshot of 2-level Model Used in Sanity Test Study

Note that we could easily have found parameters to bring recognition accuracy in either experiment, in particular in the 40% noise experiment, arbitrarily close to 100%. E.g., simply raising K from 5 to 6, which would increase run times slightly, would likely achieve accuracy $\sim 100\%$. Also, although our particular noisification method yields frames in which the perturbed pixels move to locations adjacent to the original contour, the algorithm is not sensitive to how far the pixels are moved: the results would be essentially similar for salt-and-pepper noise, and in fact for more structured perturbation, e.g., displacing/transforming whole segments of contour.

Table 3. Performance of 2-level Model on Weizmann Edge Snippets

Common Properties Across Experiments		
# of Macs = 20 , #CMs / mac (i.e., Q) = 4 , #cells / CM (i.e., K) = 5		
# weights (U+H)	4,391,500 (small fraction of wts increased, so probably could store several fold more snippets while still getting very high accuracy)	
# input pixels	84x120 = 10,080	
# Snippets = 90 , # Frames = 1,722		
	Test = Train	Test \neq Train”; 40% Noise
Recognition (trace) Accuracy	Ave. over all frames: 93%	Ave. over all frames: 92%
	Ave. snippet-final frames: 92%	Ave. snippet-final frames: 93%
Class Accuracy	98% (88/90)	93% (84/90)
Train Time	187 sec.	187 sec.
Test Time	160 sec.	148 sec.

5.3.2. Lower Resolution Weizmann Edge

We believed it likely that the Weizmann classification task could be accomplished with substantially lower spatial and temporal resolution. We therefore produced a set of Weizmann Edge snippets with half the spatial resolution, e.g., $42 \times 60 = 2,520$ pixels, and half as many frames. Figure 18 shows example frames at the lower resolution. The much smaller number of pixels comprising the input surface allowed us to reduce the number of L1 macs needed to cover the range of active pixels per frame fairly well.



Figure 18: Lower Spatial Resolution (42x60) Original and 20% Noisy Frame

As Table 4 shows, the 2-level model was able to learn the training set with very high fidelity, one trial each, in *18 sec*. For the “Test=Train” case, the test run yielded 100% classification accuracy and very high trace accuracy and took only 10 sec. Note that we can easily find parameters to bring trace accuracy very close to 100%, e.g., making the 16 macs be used more evenly, which lowers crosstalk during retrieval. However, we concluded that achieving 100% class and 97% trace accuracy in the “Test=Train” (sanity) case is sufficient to move on to other tasks/experiments.

Table 4. Model Performance on Low Space-Time Resolution Weizmann Edge Snippets

Common Properties Across Experiments		
# of Macs = 16 , #CMs / mac (i.e., Q) = 4 , #cells / CM (i.e., K) = 5		
# weights (U+H)	920,000 (small fraction of H wts increased, so probably could store several fold more snippets while still getting very high accuracy)	
# input pixels	42x60 = 2,520	
# Snippets = 90 , # Frames = 883		
	Test = Train	Test \neq Train”: 40% Noise
Recognition (trace) Accuracy	Ave. over all frames: 89%	Ave. over all frames: 97%
	Ave. snippet-final frames: 97%	Ave. snippet-final frames: 99%
Class Accuracy	100% (90/90)	100% (90/90)
Train Time	18 sec.	17 sec.
Test Time	10 sec.	10 sec.

We also produced various noisy versions of the low space-time resolution Weizmann data. We produced two types of noisy data, a) “structured” noisy data in which, for each frame, we moved a randomly chosen subset of pixels but moved them to nearby locations (see Figure 18a); and b) salt-n-pepper noisy data, where we moved randomly chosen pixels to arbitrarily distant locations in the frame (no figure shown). For each type of noise, we produced 10, 20, and 40% noisy versions of the snippets. The model performs essentially equally well on both types of noise and so we only report results for the 40% salt-n-pepper noise condition in Table 4. As can be seen, the model performs nearly perfectly for this condition and the test time is the same as for the sanity test, 10 sec. Again, regarding processing speed, we emphasize that in addition to the fact that no machine parallelism is used here, we expect we could achieve at least 10x-100x speed-up via standard algorithm and software optimizations.

5.3.3. Sanity Test: 3-Level Model Revealing DCCI Principle

Here, we show that a 3-level Sparsey can achieve the same *classification* performance as the 2-level model in the “test=train” condition. Comparing Table 5 to Table 4 and Table 3 shows that the trace accuracy achieved by the 3-level model is substantially lower than the 2-level model. However, this is in fact not a problem. In fact it reveals a key principle:

Differential correlation of correctly vs incorrectly active cells with correct classifications (DCCI, “differential correlation of correct and incorrect”): When many macs are involved in the class decision, the codes in those macs can contain many errors (i.e., low trace accuracy) while still supporting high class accuracy. This is because the correct cells across all active macs influencing the class decision are highly correlated with the correct class cell, whereas the incorrect cells across those macs are far less correlated. Thus, the input summation of the correct class cell from those macs will tend to rise above the summations of all the other (incorrect) class cells.

Looking at Table 5, one can see that for all cases where $K > 1$, trace accuracy (the R measures) can be quite low, e.g., 40%, while still yielding nearly 100% class accuracy.

Table 5: 3-Level Model Performance in “Train = Test” Condition

# Snippets = 90, # Frames = 883, L1 has $M_1 = 468$ macs (18x26), L2 has $M_2 = 192$ macs (12x16)														
	Train Time (sec)	Test Time (sec)	L1				L2				Wts (10^6)	R^*	R^Ω	Class Acc.
			Q	K	R^*	R^Ω	Q	K	R^*	R^Ω				
1	112	37	6	6	0.43	0.40	5	6	0.47	0.46	9.97	0.45	0.43	0.99
2	124	30	5	6	0.43	0.39	5	6	0.49	0.46	7.86	0.46	0.43	0.98
3	90	21	4	6	0.43	0.39	4	6	0.46	0.43	5.11	0.44	0.41	0.98
4	60	18	4	5	0.45	0.41	4	5	0.49	0.48	3.62	0.47	0.45	1.00
5	49	16	4	4	0.46	0.43	4	4	0.53	0.51	2.39	0.50	0.47	0.98
6	34	14	4	3	0.51	0.48	4	3	0.60	0.59	1.41	0.56	0.54	1.00
7	27	13	4	2	0.62	0.60	4	2	0.70	0.69	0.68	0.66	0.65	0.97
8	16	11	3	2	0.63	0.61	3	2	0.70	0.69	0.41	0.66	0.65	0.97
9	13	10	3	1	1.00	1.00	3	1	1.00	1.00	0.13	1.00	1.00	0.96
10	12	9	2	2	0.62	0.61	2	2	0.70	0.69	0.21	0.66	0.65	0.92
11	10	8	2	1	1.00	1.00	2	1	1.00	1.00	0.07	1.00	1.00	0.96
12	9	8	1	1	1.00	1.00	1	1	1.00	1.00	0.03	1.00	1.00	0.96

The goal of the experiments in Table 5 was to see how small we could make the model (how few weights it could have) while still passing the sanity test. As is clear, we were able to reduce the size of the macs all the way down to the localist condition, i.e., each mac had only one cell, while still achieving nearly 100% accuracy. In that condition (row 12), there were $< 30,000$ weights and the model learned in 9 seconds (again, standard software optimizations could probably reduce this by 10-100x).

Figure 19 shows the 3-level model used in these experiments. It has 468 L1 macs and 192 L2 macs. The bottom-up receptive fields, U-RFs, of several L1 and L2 macs are shown (cyan patches). Active macs are shaded rose unless they fall within a depicted RF (“RF” is used for “U-RF” in the figure since there is no ambiguity) in which case they are shaded purple. For example L1 mac A’s U-RF consists of ~ 40 pixels (the darker cyan patch falling within the U-RF of L2 mac X).

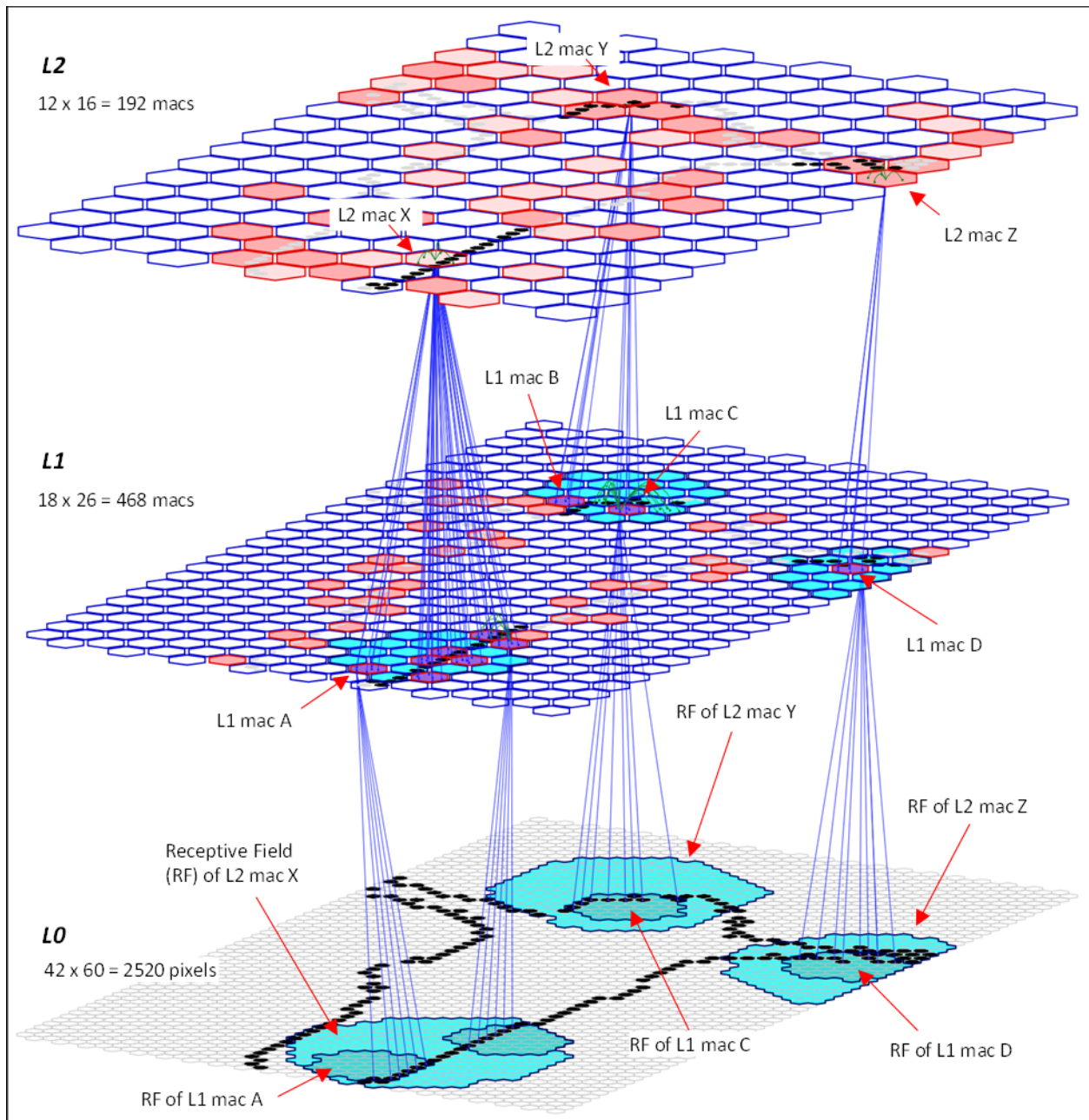


Figure 19: Model Achieving ~100% Class Accuracy in Weizmann Sanity Test

We make the following points regarding Figure 19:

1. Any mac at any level represents a particular *part*, or feature, of the whole. Each such part is a composition of several lower-scale parts represented at the subjacent level. Three L2-scale parts are shown superimposed over the L2 macs that represent them (black pixels). Mac X is actively representing a vertical edge segment, Y is representing an upper left rounded corner, and Z is representing a less canonical looking feature.

2. Though not shown here, in general, the U-RFs of neighboring macs at any level have substantial overlap. The *immediate* U-RF of a mac at L2 or higher consists of a patch of macs in the subjacent level. We refer to the U-RF at level J of a mac, m , at a higher level, as $\text{U-RF}(m, J)$. We show three multi-level-spanning U-RFs, $\text{U-RF}(X, 0)$, $\text{U-RF}(Y, 0)$, and $\text{U-RF}(Z, 0)$, at L0.
3. There is generally substantial pixel-level overlap between the parts represented by nearby active macs. This is one form of overcompleteness present in Sparsey; we call it *inter-mac overcompleteness*. There is also *intra-mac overcompleteness*: the input patterns represented by the different codes stored in any particular mac will also generally have a substantial pixel-level overlap. An additional nuance that must be appreciated is that for macs at L2 and higher, whose U-RFs are sets (patches) of subjacent macs, we can also define overcompleteness in terms of mac-level overlap (in contrast to pixel-level overlap).
4. One can see that there are seven active L1 macs in L2 mac X's U-RF. Parameters determine the range of active macs in a mac's U-RF for which it will become active. We tile these π -bound parameters over the mac sheet so as to ensure, probabilistically, that any particular active feature (either a pixel or a mac) at any level is likely to fall within the U-RF of at least one active superjacent mac.
5. The U-RFs of several L1 macs are also shown at L0. These are the slightly darker cyan patches falling within the larger patches. It is the portions of the input that falls within these L1 mac U-RFs that are causing those macs to become active. It can be seen, most clearly within $\text{U-RF}(X, 0)$, that parts represented by L1 macs are smaller than those represented by L2 macs. In fact, two disjoint parts corresponding to two L1 macs, A and one other, are clearly seen. The part represented by X is the union of these two parts and those represented by the other five active L1 macs in X's immediate U-RF (to reduce clutter, their U-RFs are not shown).

Figure 20 shows another example, from another frame of another snippet, illustrating the richness of Sparsey's modular, i.e., mac-based, representation of the hierarchical part-whole structure of inputs. At top, we show the portion of the input pattern represented by an L2 mac (with blue lines leading to it) superimposed over it (black pixels). This "whole" is naturally describable as consisting of three parts, e.g., as "three roughly parallel roughly vertical edge segments spread out horizontally and rising slightly toward the right". One could imagine it being a potentially useful feature/part that could appear as a part of many possible larger images. The representations of the three constituent parts can be seen at L1. One can see that the L1 mac's have substantially overlapping U-RFs. Thus, four L1 macs (purple) L1 macs actually participate in representing those three parts: the two parts associated with the middle L1 macs are highly overlapped and appear as one.

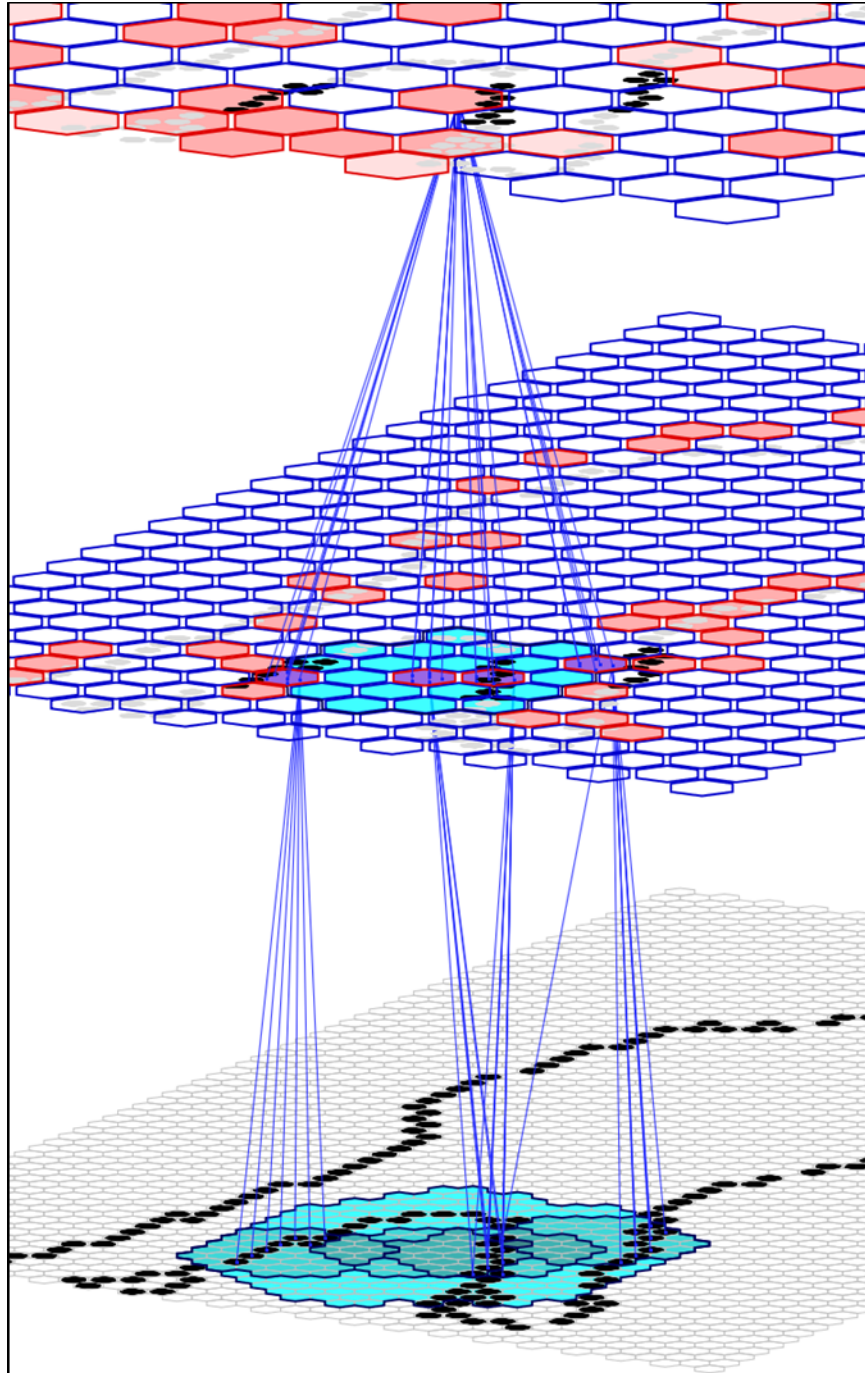


Figure 20: Example Showing Richness of Sparsey's Modular, i.e., Mac-Based, Representation of the Hierarchical Part-Whole Structure of Inputs

5.4 Family-Resemblance Classification Style

It has long been appreciated that precise definitions of natural objects and events in terms of lists of necessary and sufficient features (or parts) are easy to break. This underlies the “brittleness” of classical, logic-based AI systems. Natural categories are better described in terms of *family resemblances*, i.e., lists of features (parts) *associated* with instances of a category but no one (or any small subset) of which is absolutely necessary (Wittgenstein 1953). In terms of Sparsey, a family-resemblance based classification style is revealed to the extent that different though overlapping sets of top-level macs are active as the representation of different instances of the same class, an effect revealed in this experiment.

We tested the 3-level model of Figure 19 (with parameters set as in row 2 of (Table 5) on 20% noisy versions of the training set. Table 6 reports the results of a subset of these numerous experiments while searching for good parameter configurations. In Experiment 1 (row 1), we presented only 20 of the 90 snippets (two exemplars of each of the ten classes). Class accuracy was 80% (16 out of 20). The very interesting point here is that the trace accuracy is very low at both L1 (~15%) and L2 (~13%). This is strong evidence of the DCCI principle described above. In Experiment 2, we ran the same model on all 90 snippets. The class accuracy dropped substantially due to the increasing effects of crosstalk, but still remained far above chance (chance=10%).

We noticed that the set of active snippet-final, top-level macs (i.e., the macs whose outputs drive the classifications) differed substantially across exemplars of the same class. We see this even in comparing a training snippet with a 20% noisy version of itself, as shown in Figure 21. There are many active (rose) L2 macs in common between the left and right panels, but there are also many differences. The same is true for the L1 macs as well. Since we see this even when one snippet is a noisy version of the other, we certainly expect to see this effect when comparing training snippets to novel test snippets of the same class. Since the L2 macs drive the classifications, we need to understand principles by which substantially different sets of L2 macs, as well as substantially different codes in those macs, can learn to activate the same class cell.

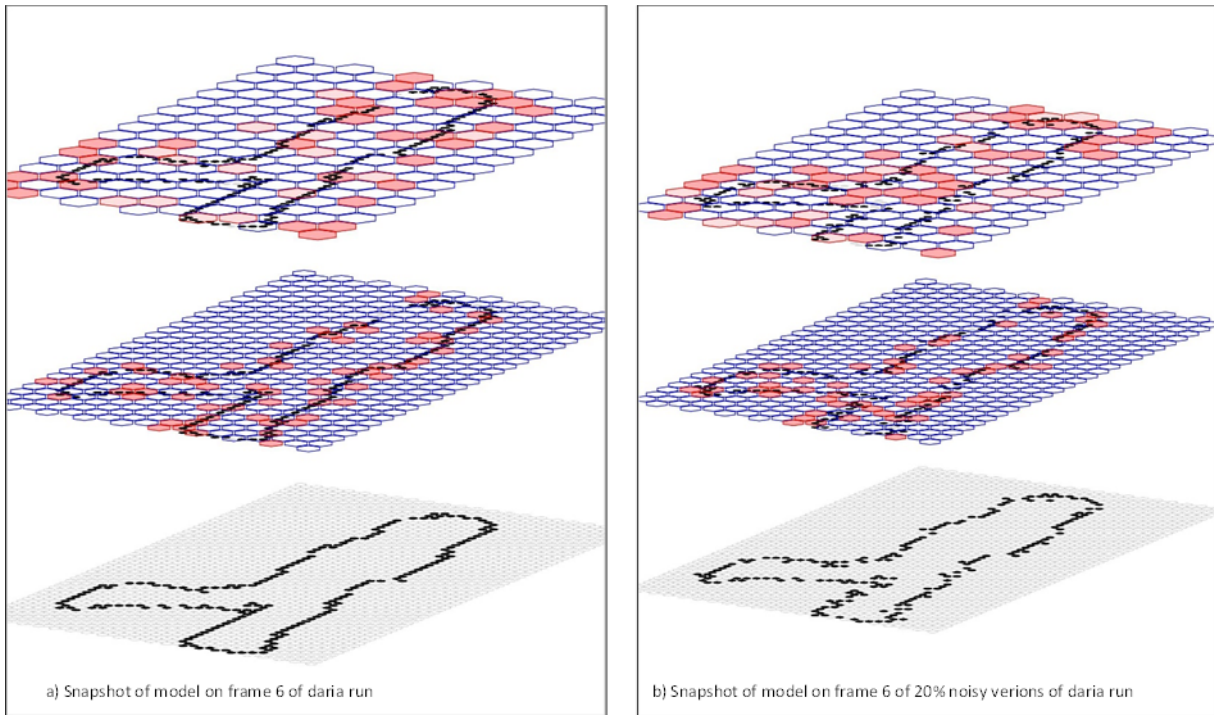


Figure 21: Substantially Different Though Overlapped Sets of Active L1 and L2 Macs Between Original and Noisy Version of a Snippet

In some sense, one goal of our parameter search must be to find parameters that result in greater similarity over the set of all snippet-final top-level active mac sets within any given class (and thus, averaged across all classes). All else equal, this increases the separability of the inputs and should increase performance. In order to achieve this, it is clear that the macs need to be somewhat lenient in their *match* [more specifically, *familiarity* (G)] computation during test. Accordingly, a principal goal guiding the parameter search underlying the results in Table 6 was to maximize classification accuracy, by increasing the leniency of the spatiotemporal match. There are many parameters that influence this: the normalization thresholds applied to each of the H, U, and D, inputs; the sharpening (raising to a power), also applied to the individual inputs prior to mixing, and the parameters of the sigmoid during learning, and others. Table 6 presents a few data points in this search process. General principles about the needed relationships of the parameters are still emerging.

Table 6. 3-Level Model Performance in “Test = 20% Noisy Train” Condition

	Num. Frames	L1					L2					R*, R ^Ω	Class Acc.
		Sharp. exps U,H,D (train, test)	Norm max cutoff U,H,D (train)	Norm max cutoff U,H,D (test)	Sig. upper V cutoff	R*, R ^Ω	Sharp. exps U,H,D (train, test)	Norm max cutoff U,H,D (train)	Norm max cutoff U,H,D (test)	Sig. upper V cutoff	R*, R ^Ω		
1	208	2,2,2 2,2,2	0.9, 0.9, 0.9	0.9, 0.9, 0.9	0.99	0.15, 0.14	3,3,2 3,3,2	0.9, 0.9, 0.9	0.9, 0.9, 0.9	0.99	0.11, 0.11	0.13, 0.12	0.80 16/20
2	883	2,2,2 2,2,2	0.9, 0.9, 0.9	0.9, 0.9, 0.9	0.99	0.12, 0.11	3,3,2 3,3,2	0.9, 0.9, 0.9	0.9, 0.9, 0.9	0.99	0.09, 0.09	0.11, 0.10	0.52 47/90
3	208	1,2,2 1,2,2	0.95, 0.95, 0.95	0.8, 0.8, 0.7	0.3	0.15, 0.16	1,2,2 1,2,2	0.95, 0.95, 0.95	0.8, 0.8, 0.7	0.3	0.13, 0.12	0.14, 0.14	0.65 13/20
4	208	2,2,2 1,2,2	0.95, 0.95, 0.95	0.8, 0.8, 0.7	0.3	0.15, 0.15	1,2,2 1,2,2	0.95, 0.95, 0.95	0.8, 0.8, 0.7	0.3	0.13, 0.13	0.14, 0.14	0.55 11/20
5	208	2,2,2 1,2,2	0.95, 0.95, 0.95	0.4, 0.8, 0.7	0.3	0.13, 0.13	1,2,2 1,2,2	0.95, 0.95, 0.95	0.8, 0.8, 0.7	0.3	0.13, 0.13	0.13, 0.13	0.60 12/20
6	208	2,2,2 1,2,2	0.95, 0.95, 0.95	0.8, 0.8, 0.7	0.3	0.15, 0.15	2,2,2 1,2,2	0.95, 0.95, 0.95	0.8, 0.8, 0.7	0.3	0.12, 0.12	0.14, 0.13	0.65 13/20
7	208	2,2,2 1,1,1	0.95, 0.95, 0.95	0.8, 0.8, 0.7	0.3	0.16, 0.15	2,2,2 1,1,1	0.95, 0.95, 0.95	0.8, 0.8, 0.7	0.3	0.12, 0.12	0.14, 0.14	0.65 13/20
8	208	2,2,2 1,1,1	0.95, 0.95, 0.95	0.8, 0.8, 0.7	0.8	0.13, 0.12	2,2,2 1,1,1	0.95, 0.95, 0.95	0.8, 0.8, 0.7	0.8	0.10, 0.10	0.12, 0.11	0.70 14/20

5.5 Experiments with More Powerful Input Feature, HOFs

In the May-June 2015 timeframe, we were encountering difficulty tuning parameters of multi-level Sparsey models to perform well on classification of the *edge*-filtered versions of the KTH snippets. We therefore decided to test Sparsey using more powerful inputs features, in particular, the motion features known as HOF (Laptev, Marszalek et al. 2008). We point out that all SOA results on video event classification benchmarks have been achieved using motion features, e.g., HOF, MBHs, etc.

We implemented a preprocessor that produces HOF-filtered versions of video data sets. To increase our chance of success, we also switched from using KTH snippets, to the slightly easier Weizmann snippets. However, we still encountered difficulty achieving SOA classification performance on Weizmann. Not only that, our multi-level model was also having difficulty getting high scores on the “sanity test” baseline, i.e., in which the test set is the same as the training set. We therefore decided to remove the hierarchical parameter tuning problem from the equation by scaling back to a 2-level model (i.e., having only one internal level of macs). This led immediately to achieving nearly 100% accuracy (in terms of both recapitulating the exact spatiotemporal activation traces and outputting the correct class/label) in the “train=test” condition. Figure 22 shows the 2-level model that achieved 97% recognition accuracy on all 90 HOF-filtered Weizmann snippets, which were presented once each. This [web page](#) contains a concise description, but with videos, of this model and the HOF input data format. We note the following properties of this model:

1. The single internal level contains 20 macs
2. These are small macs, having only $Q=5$ CMs, and each CM having only $K=7$ cells. Thus, each mac code, a SDC, is just a set of 5 active cells.
3. The bottom-up receptive field (U-RF) of every mac is the same and includes the entire input level ($24 \times 120 = 2,880$ binary pixels). Of course, these binary “pixels” represent presence/absence of motion in a small number of canonical directions at a particular locale of visual space. See [this page](#) for more description of the input format.
4. The horizontal receptive field (H-RF) of every mac includes the other 19 macs.
5. Each mac has been set to activate for a different range of active (black) pixels. E.g., the upper left mac activates if between 5 and 14 (out of the entire 2,880 pixel input surface) pixels are active, the next one if between 15 and 28 pixels are active, and so on, down the lower right mac, which activates if between 389 and 450 are active. By assigning disjoint activation ranges, we ensure only one mac is active on any frame. This: a) makes it easier to see and understand what the model is doing, i.e., building spatiotemporal chains of SDCs as representations of spatiotemporal input patterns (sequences); and b) minimizes the rate at which wts (both U and H) are increased during learning, which indirectly increases the number of such sequences that can be learned (i.e., storage capacity).
6. Figure 22 shows all U (blue) and H (green) wts leading to the active (black) cells in the active (rose) mac that were increased at any time during learning. Despite appearances, only a small fraction of the model’s overall wts were increased during learning, suggesting the

model could probably store a great deal more such snippets while still achieving very high recognition and classification accuracy.

7. By items 4 and 5 above, we get simple chains of mac activations activating over the course of the snippet and such that on all frames except the first frame of each snippet, a mac becomes active in the context of both U inputs and H inputs, i.e., in a spatiotemporal context. As explained in earlier reports (and throughout my work) this makes the codes selective to those spatiotemporal contexts and reduces confusion during recognition, enabling better accuracy. Figure 23 shows an example of a 4-frame HOF input (produced from about 30 original frames of the “Ido_Walk” Weizmann snippet) and its spatiotemporal memory trace embedded in the mac level. U (blue) and H (green) wts that have been learned in the past and are coming from currently active pixels (in the case of U wts) or from previously active internal cells (in the case of H wts) are shown. Note that any given mac can become active multiple times during the snippet, e.g., the same mac is active on frames 2 and 4, though in general, the code will be different on different occasions (as is the case here).

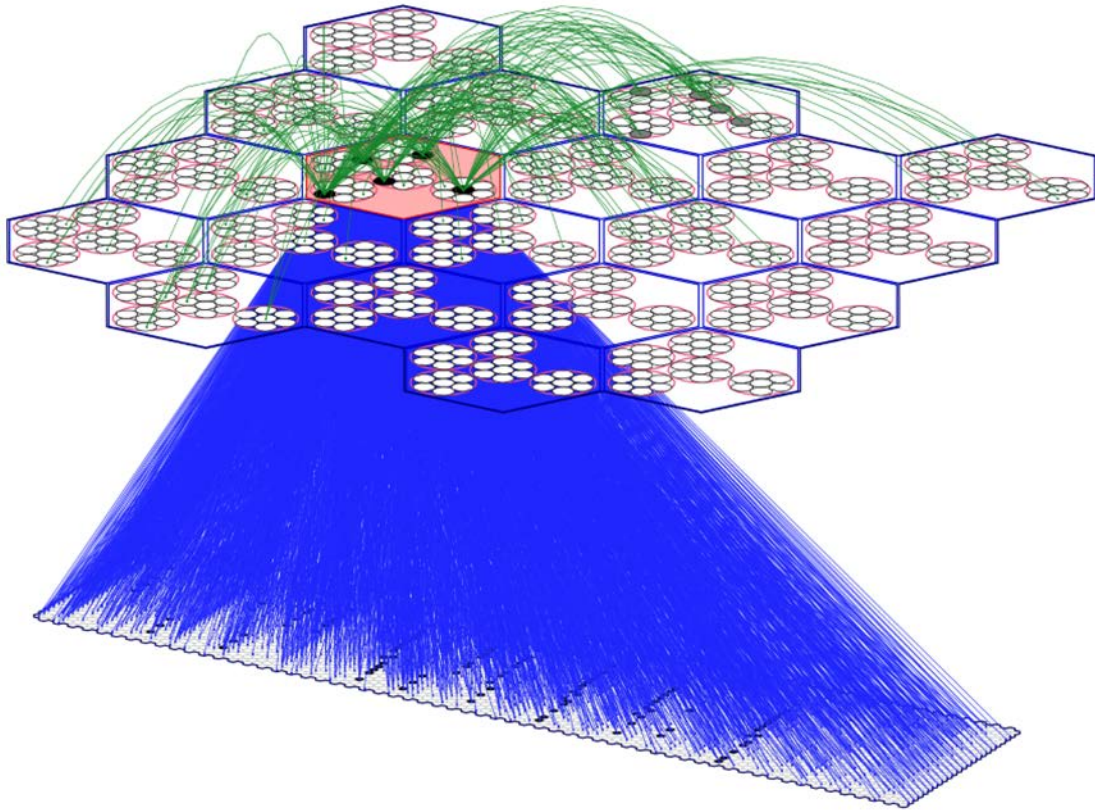


Figure 22: Model Achieving ~100% Accuracy on HOF-filtered Weizmann Dataset

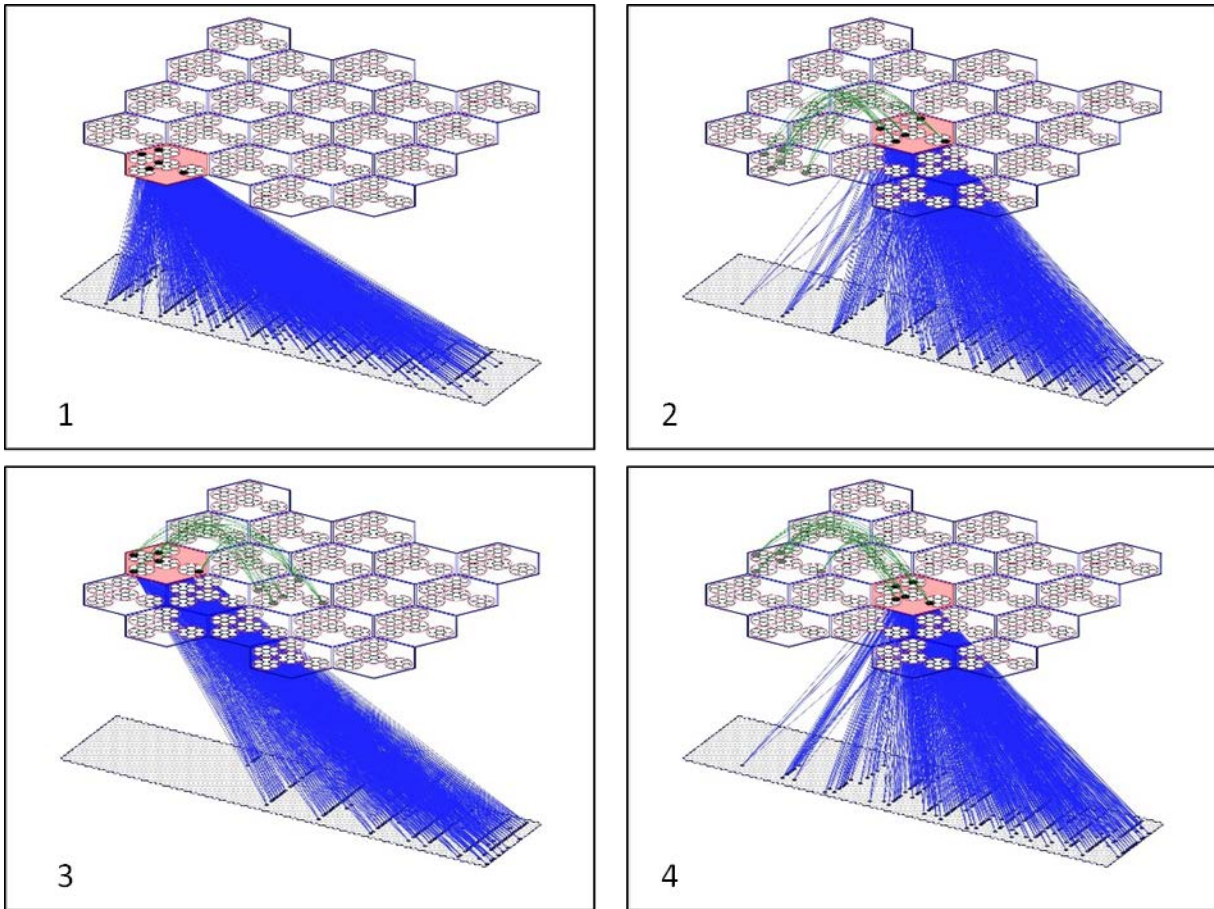


Figure 23: 4-Frame HOF Snippet and Corresponding Spatiotemporal Memory Trace

This model's performance on the sanity test condition is given in Table 7. It achieved 97% class accuracy and took only 25 seconds to train on a single 3.2 GHz processor. Note that minor variations of some of the parameters, led to many runs that achieved 99% accuracy.

Furthermore, for many reasons, the train time can probably be greatly reduced. Notably, the input surface is large (2,880 pixels) and the HOF vectors produced have huge redundancy. It is very likely that we can find an input representation that is far smaller and at least as easily learned. In addition, we believe at least 10x-100x speed-up is possible just from more thorough, but standard, algorithm/code optimizations.

Table 7. Performance of 2-level Model on “Test = Train” Condition

# of Mac	20
# CMs / mac (Q)	5
# cells / CM (K)	7
# weights (U+H)	2,529,065 (small fraction of wts increased; likely could store several fold more snippets and maintain very high accuracy)
# input pixels	2,880
# Snippets	90
# Frames	547
Recognition (trace) Accuracy	Ave. over all frames: 90% Ave. of last frames of snippets: 96%
Class Accuracy	97% (87/90)
Train Time	25 sec. (single core, 3.2 ghz). Likely at least 10x-100x speed-up with standard algorithm/software optimizations.
Test Time	15 sec.

Having demonstrated that Sparsey can pass the sanity (test = train) case of the Weizmann data set, our next step was to demonstrate that it could pass the actual benchmark (test \neq train) Weizmann task. However, we still had the following problem. As is apparent comparing the nine frames of a snippet shown in the first column of Figure 24, our HOF vectors are large (relative to the amount of information likely needed to disambiguate these event classes) and highly redundant. Moreover, it appears that the within-class variance over the inputs is as large as the between-class variance, which makes learning a classification hard. Therefore, before moving to the full “test \neq train” case, we decided to first verify that Sparsey could recognize noisy instances generated from the training set.

We created noisy versions of our HOF-filtered Weizmann data set. Figure 24 column two shows a 40% noisified version of one of the original snippets (“Ido_Bend”) shown in column one. We took each frame of each snippet and randomly changed $x\%$ of the active pixels, for 10, 20 and 40%. [This web page](#) shows the original and noisy movies corresponding to another snippet (ideally, the two snippets should be synchronized; sometimes hitting the reload button does that). Note that since each of the resulting noisy snippets was produced from a single original snippet, we know that the noisy snippet will with probability close to one be much more similar (in terms of Hamming distance) to the snippet from which it was produced than to any other (either within its own class or across classes). Therefore, the class of the noisy snippet will be the same as the class of the original from which it was produced.

Despite this substantial difference between the original and noisy snippets, the Sparsey model of Figure 22 achieved 96% accuracy in classifying even the 40% noisy data set. Specifically, we created 5 different instances of the 40% noisy data, resulting in 450 test instances, 45 of each of the 10 classes; the model got 432 of the 450 correct. We note also that the model parameters were actually set to enforce rather stringent spatiotemporal similarity metric in this experiment and that less stringent settings easily achieves close to 100% classification accuracy.

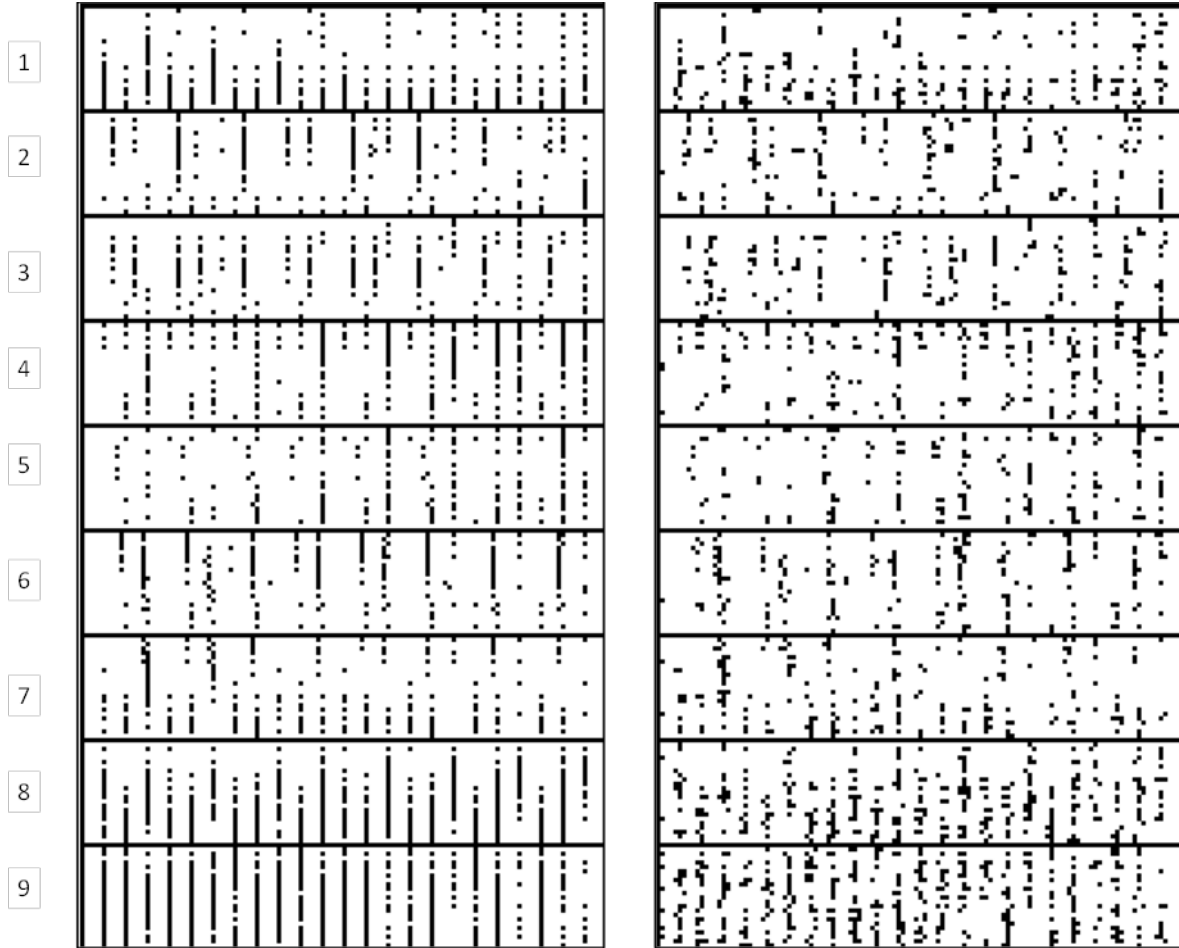


Figure 24: Original and 40% Noisy Frames of the Ido_Bend Snippet

While achieving such high tolerance to this “salt and pepper”-type noise does not demonstrate learning the underlying structural regularities (in space and time) that define these high-level event classes, it does demonstrate that Sparsey can assign spatiotemporally more similar input patterns to more similar internal representations (SDCs, and chains of SDCs). We were encouraged by the noisy test results since, at some level, this SISCs property must be true of any successful recognizer of any class. However, when we carried out the full “test \neq train” experiment, the model did not attain high classification accuracy.

At this point in the research, we had a choice to make as to whether to continue using the HOF features or return to using edge features. We decided on the latter path for three reasons:

1. Increasing our success with the HOF features would have required a significant investment of time to cause our HOF-generating preprocessor to generate far less redundant and far more compressed codes that we believe are needed to increase classification performance. In contrast, our edge-filtered inputs already have low redundancy even when we substantially compress them.

2. Regardless of which feature set we would use, we knew that we still required significant exploration of the model’s very large parameter space. In particular, any realistic final model capable of solving any applied video event recognition task will consist of multiple levels. As we have noted throughout the project and as those at the leading edge of the field know, hierarchical decomposition of the input/problem space is essential to scalability and generalizability. We faced the hurdle of understanding the correct parameter relations across levels regardless of the feature type used. Thus, this criterion did not favor either approach.
3. Sparsey’s overall concept of operations (CONOPS) is that it will *automatically* be able to discover *all* higher-level features of the input space, both spatial and spatiotemporal, from scratch. That is, it will be able to build low-level motion primitives, e.g., translating edges, rotating edges, directly from sequences of purely spatial features, i.e., the pixels. And, it will be able to scaffold this same principle up through multiple levels (spatiotemporal scales). This last criterion argued strongly to return to using edges.

For those reasons, we decided to return to using edge features. While it is possible that we may have been able to achieve higher classification accuracy on the benchmark Weizmann task more quickly using HOF features, we believe that the fact that we are close to achieving SOA classification levels on Weizmann vindicates our choice. That said, we have completed a great deal of infrastructural work and experimentation using HOF features and we will complete our planned bi-modal fusion, “edges + HOFs” investigations (see Figure E-1) in the near future.

5.6 Principles/Mechanism of Invariance

A main goal of our research has been to understand how whole Sparsey networks, consisting of many levels and hundreds/thousands of macs, can tolerate not only noise but more structured perturbations, e.g., omissions/insertions of whole parts (features), etc. In other words, what principles/mechanisms underlie Sparsey’s ability to learn the kinds of large-scale, highly nonlinear invariances that necessary to perform well on any of the existing video data sets (KTH, UCF, etc.)? We describe some of the essential principles/mechanisms in this section and the associated appendices.

We have seen numerous instances in which a mac at a given level, J , (we’ll refer to this mac as the *target* mac) receives input from several level $J-1$ macs (the *source* macs), some of which have low G values (indicating poor recognition of their inputs), but nevertheless achieves high, often 100%, recognition accuracy. Given that each of these active source macs represents the presence of a particular feature, such instances indicate the target mac is recognizing novel inputs *invariant* to often large deformations, even complete omissions or intrusions, of individual features. Figure 25 shows one such example.

Figure 25a depicts the overall situation in which L2 mac M_{402}^2 activates in response to input in its receptive field (outlined in yellow in bottom level). The inset at left places this example in the context of the whole 9-level network. The yellow ellipse in L1 shows the set of 10 L1 macs comprising M_{402}^2 ’s immediate RF, two of which are active (purple), M_{773}^1 and M_{813}^1 (this RF is also more clearly visible in inset at upper left of panel c). To understand this example, we must compare the situation in panel b, where the same input frame has been presented as on the

training trial (“Test = Train”) to that of panel c, where a noisy version of that input frame has been presented (“Test \neq Train”). Panel b shows that when the original training input was presented, L1 macs M_{773}^1 and M_{854}^1 activated. This is because they each had an appropriate number of active pixels in their U-RFs. M_{773}^1 ’s U-RF is outlined in green in the box adjacent to its tree node and M_{854}^1 ’s is outlined in red in the box next to its node. At upper right, we show how these two U-RFs overlap and where they fall within M_{402}^2 ’s larger U-RF. Note that mac M_{813}^2 is not active (its node is gray and connector is red) because it has too many active pixels in its U-RF (outlined in purple). Thus, in panel b, M_{402}^2 activates on the basis of its inputs from M_{773}^1 and M_{854}^1 , each of these can be viewed as representing the presence of the particular feature, or *part*, shown in their respective U-RFs. Note further that these two parts have some common pixels.

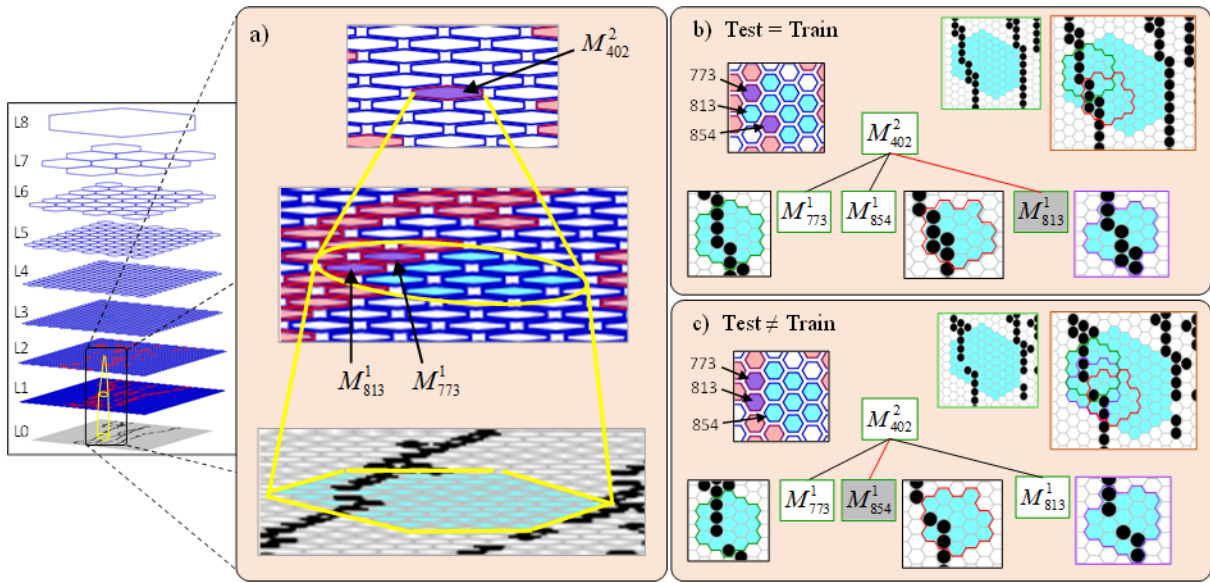


Figure 25: Demonstration of Invariant Recognition

To complete the example, we now consider panel c. The small green-bordered inset at top middle of panel c shows the noisy input in the vicinity of M_{402}^2 ’s U-RF. One can see how it differs from original training input by comparing this to the analogous panel at top middle of panel b. Because of how the noisy pixel pattern falls upon the U-RFs of the L1 macs, in general, a different subset of M_{402}^2 ’s L1 macs may activate than did for the original input. Furthermore, even if a given L1 mac activates in both instances, the exact pattern of active pixels in its U-RF may differ. As it happens, in this example, both of these apply. First, M_{854}^1 ’s U-RF no longer has enough active cells to activate (it has only 3). Second, M_{813}^1 ’s RF now has an appropriate number (5) of active pixels and so it activates. Finally, note that although M_{773}^1 activates in both cases, its active pixel patterns differ (slightly).

Thus, M_{402}^2 has what can only be considered a *very* different pattern of inputs, as mediated by L1, for the noisy input versus the original training input. Yet, the same exact SDC, activates in M_{402}^2 in both cases (code not shown). Thus, M_{402}^2 responds *invariantly* to the two inputs. There are several mechanisms/principles underlying this invariant recognition capability, the two most important probably being:

1. We use the max- V version of the CSA. That is, we simply pick the cell with the max V in each of a mac's CMs as winner. As long as the crosstalk between SDCs stored in a mac remains low enough, this allows the exact same cell to win despite wide variation in the exact V distributions within any given CM, and thus allows reactivation of the exact same overall SDC across wide variation in the mac's inputs. In fact, we provide a detailed description of this, with respect to the example of Figure 25, in Appendix D.
2. Even though whole afferent macs (and so the features that their active codes represent) might be omitted or inserted in the test trial relative to the training trial, the fact that those macs' U-RFs will generally overlap (as is the case for M_{773}^1 , M_{813}^1 , and M_{854}^1 , in our example (see upper right insets in Figure 25b,c) mean that sub-features (in this case, pixels) mediated by one afferent mac will not necessarily drop out even if that mac fails to activate because they are also mediated (redundantly) by other macs. To quantify this phenomena, we developed a new measure, "input accuracy (IA)", which measures how similar a mac's inputs are, in a way that takes account of omitted/inserted afferent macs. Preliminary results involving IA and its relation to *recognition accuracy* (R) in the target mac are given in Appendix E.

We emphasize that this invariant recognition capability is not programmed into Sparsey in any ordinary sense. That is, many past approaches build in certain large-scale invariances from scratch, e.g., log-polar-Fourier transform, to achieve scale-position-rotation invariance. While powerful in some applications, it has long been noted that such approaches are blind to the part-whole (compositional) structure of inputs, e.g., to the possibility that different parts of a whole entity can have different positions, scales, and rotations, etc., relative to each other and to the whole, and even be missing or inserted relative to other instances. The degree of invariance shown in the example of Figure 25 is perhaps not too impressive. After all, if you blur your eyes (low pass filter), the original and noisy input patterns to M_{402}^2 (green-bordered insets at top middle of Figure 25b,c) look very similar. However:

1. The mechanism giving rise to this invariance is extremely low-level (imposes only a very weak prior on the input space) and learned, not programmed.
2. The same mechanism operates simultaneously at all scales (levels) of the hierarchy.

Figure 26 summarizes the example of Figure 25, emphasizing the very different patterns of L1 activation that arise in response to re-presentation of original learned input, X (panel a), vs. presentation of a noisy version, X' , (panel b). For X , L1 macs M_{773}^1 and M_{854}^1 attain G values of 1.0 and the original codes (assigned during the learning trial of X) are reinstated perfectly, i.e., $R=100\%$, which causes M_{402}^2 to be perfectly reinstated. In contrast, for X' , M_{773}^1 is reinstated perfectly, M_{854}^1 fails to activate at all, while another mac, M_{813}^1 , which was not active for X , also activates. Despite these large differences *in its immediate inputs from L1*, the same code

activates in M_{402}^2 as did for X, demonstrating a quite robust invariance of M_{402}^2 to its immediate inputs. It may seem counterintuitive that the intermediate codes, at L1, are more widely separated than the inputs themselves. However, we are finding that the tolerance to input-level differences increases with level, as the next two examples will show.

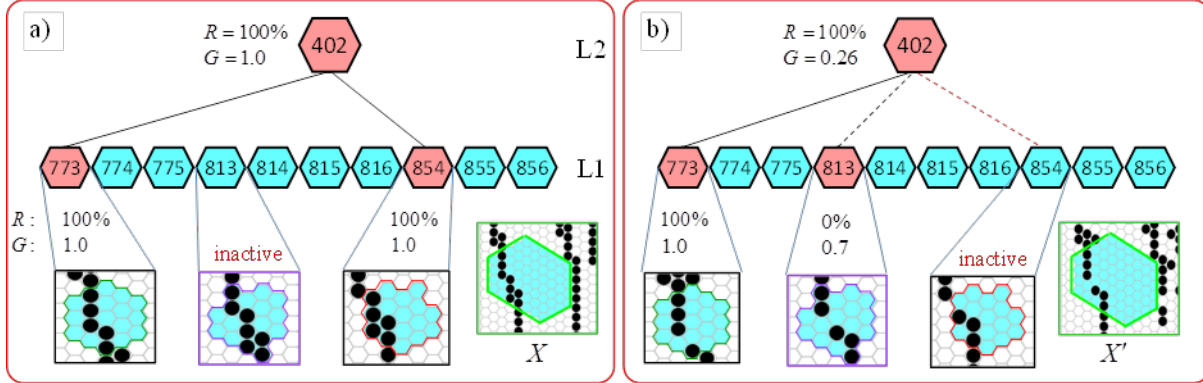


Figure 26: Demonstration of Robust Invariance

We chose these particular examples because they suggest how large the invariances can be. Broadly, it seems that most of these cases could be described as *parts* either being present in the original learned input but not in the noisy version (red ellipses), or vice versa (green ellipses). Thus, the dynamics of whole hierarchies of numerous macs appear to be realizing what looks like a quite general form of *part-invariant* recognition. And, because the various parts of an object go in and out of view as an object undergoes transforms in 3-space, part-invariant recognition is closely related to *view-invariant* recognition. We are still in the midst of building our understanding of how the invariances displayed by individual macs work together, across numerous macs and many levels, and across frames, to account for the types of large-scale invariances present in human object/event recognition. However, we are very encouraged by the robust part-invariance displayed by individual macs.

5.7 Application to Purely Spatial Pattern Recognition Problems

We have implemented supervised learning in the Sparsey simulation platform. We are currently conducting experiments with the Mixed National Institute of Standards and Technology (MNIST) handwritten digit database, i.e., with purely spatial images. We pre-processed all the original 28x28-pixel grayscale MNIST images to 24x24-pixels, binary, and 1-pixel-wide edge filtering. Figure 27 shows some original examples from the database and our corresponding preprocessed versions. For each of the digits, 0-5, we show the first ten instances of the original 28x28 grayscale images cropped to 24x24 and the preprocessed 24x24 binary, edge-filtered images.

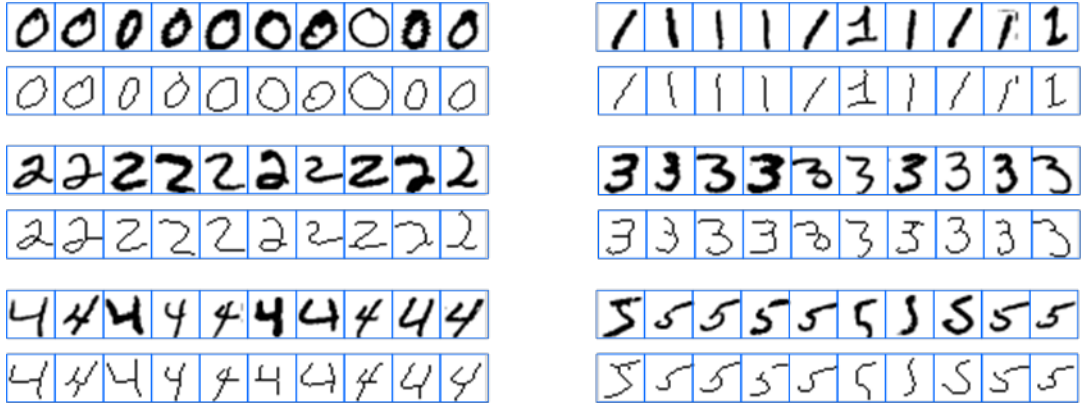


Figure 27: Sample of Original MNIST Data and Our Preprocessed Versions

We have a total of 1000 instances of each of the digits, 0-9. Our current tests, which have been for the purpose of sanity checking of various aspects of the code, getting bugs out, etc., use only 10 instances of each digit. Also, in these experiments, the test set = training set. Figure 28 shows the 10 instances of digit ‘5’. Note the substantial variability across instances. Superimposed on each digit is the grid showing the 6x6-pixel input level, “V0” (a.k.a. L0) apertures of this model. These V0 apertures connect, 1-to-1, with the V1 macs. That is, the bottom-up receptive field (U-RF) of each V1 mac is just its corresponding 36-pixel V0 aperture.

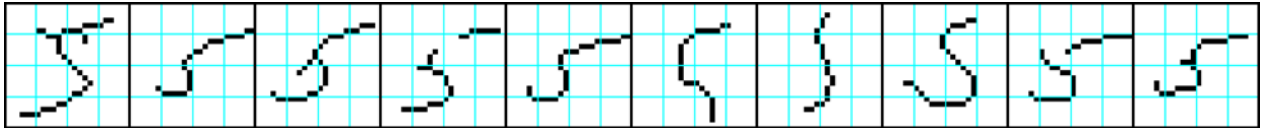


Figure 28: The 10 Instances of Digit ‘5’ Used in this Experiment

Figure 29 shows the 4-level model used with an instance of the digit ‘8’ active in V0. The first “cortical” level, V1, is a sheet of 4x4 macs. When an image is presented during training, if the number of active pixels in a V0 aperture is within a range (in this experiment, between $\pi_1^- = 5$ and $\pi_1^+ = 8$), its associated V1 mac forms an internal SDR code that represents the pattern (i.e., feature) in that aperture. Black dots indicate the active cells comprising the SDR code in each of the five active V1 macs. The next higher level, V2, consists of 4 macs (three of which are active here), each receiving input from the four V1 macs under it. The top level, V3, consists of one mac that receives input from all four V2 macs. This figure shows the recognition test trial in which the ‘8’ was presented, giving rise to a flow of U-signals activating codes in macs throughout the hierarchy, and finally a top-down flow from the activated V3 code to the Label field, where the unit with maximal D-summation, the ‘8’ unit, wins.

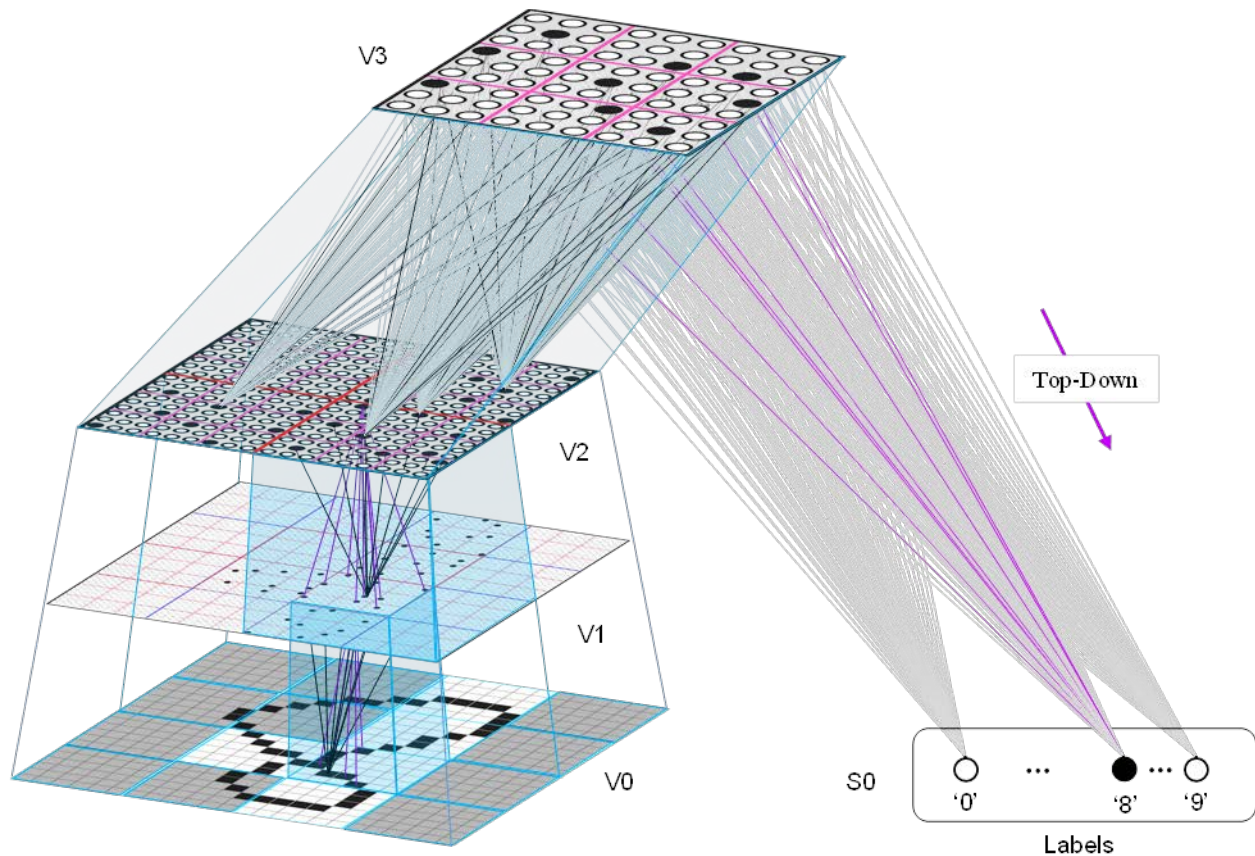


Figure 29: The 4-level Model with Preprocessed MNIST Digit ‘8’ Active in Input Level

The processing is similar in all macs at all levels: if there is sufficient activity in its U-RF, it becomes active and assigns an SDR to represent the feature present in its U-RF. The U-RF of one mac at each internal level is highlighted by a semitransparent blue prism: there is no overlap between the U-RFs of different macs at the same level, but future studies will also investigate overlapping U-RFs. The process by which an SDR code is chosen in a mac (our CSA) was described in previous reports. Once all the SDR codes throughout all the levels have been chosen, the (binary) weights between codes at adjacent levels are increased by Hebbian learning. A small sample of increased U-wts (black lines), top-down weights (D-wts, magenta), and naïve ($w=0$) U-wts (gray) are shown in Figure 29. Finally, the D-wts wts from the active V3 code to the single active unit representing the correct category (label) are increased (the U-wts from the category unit to the V3 code are also increased, but these are not needed to test recognition).

To test performance, we present the test images (again, in this experiment, the test images were the same as the training images). When an image is presented during recognition, signals propagate up through the visual levels (via the U-wts) and cause SDR codes to be re-activated in the macs at the different levels. Once the V3 code is active, the D-signals from the active cells comprising that are propagated to the label field and we activate the unit with the highest top-down (D) summation. Table 8 shows a typical result from the current experiments that we are running. The model got 75% of the test set correct. For each level, we report the average across all macs of the level and across all images in the test set of the measure, R^* , which is the

accuracy of SDR codes activated during recognition compared to those assigned during learning are reported.⁵

Table 8. Recognition Accuracy Results for Experiment 1

Num. of Test Images	Correct	Classification accuracy
100	75	75%
SDR Code Accuracy By Level		R*
L3		0.81
L2		0.79
L1		0.85
Ave. Over Levels		0.82

We are currently exploring the relationships amongst the model’s many parameters vis-à-vis various performance measures. For example, we are varying: numbers of macs per level, numbers of winner-take-all (WTA) CMs per mac, numbers of cells per CM, parameters controlling the normalization of a cell’s input summation into a likelihood of becoming active, etc. Figure 30 shows the dataset, cropped to a frame size of 16x24 and with V0 aperture size set to 4x4 pixels. The highlighted patches will be explained in conjunction with Figure 33.

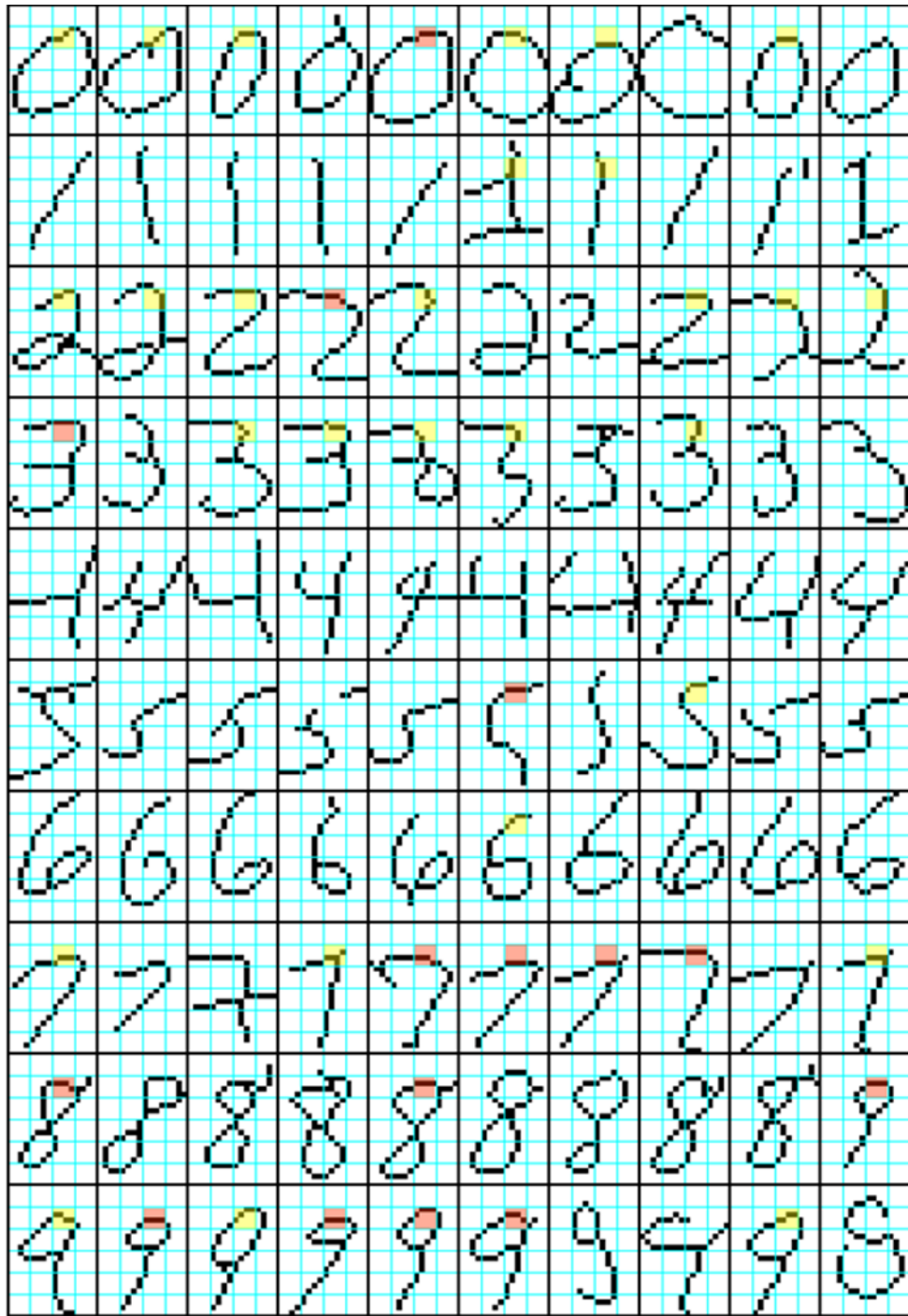
Figure 31 shows the model used in this study (except that the numbers of CMs/mac and cells/CM are larger than depicted). V1 is 4x6 array of macs, compared to the 4x4 array of macs in the model discussed in Figure 27 and Figure 28. V2 is a 2x3 array of macs, and V3 is a single mac. The blue prisms show the bottom-up receptive fields (U-RFs) of the mac at the top of the prism. The prism at lower left shows that a single V0 4x4-pixel aperture is the U-RF of a single V1 mac. Five pixels are active in that aperture. Yellow highlighted V0 apertures are ones meeting the (π_1^-, π_1^+) constraints: thus their associated V1 macs are active (black cells denote the active SDR codes in the macs).

1. The blue prism at left between levels V1 and V2 shows that the lower left V2 mac’s U-RF consists of the four indicated V1 macs. Only one of those V1 macs is active. The SDR code active in that V1 mac represents the whole 5-pixel contour active in its V0 aperture. From the standpoint of the V2 mac, it represents the presence of *one* feature in the V2 mac’s 8x8-pixel aperture.
2. The blue prism at right between V1 and V2 shows that the upper right V2 mac has two features active in its aperture.
3. The prism between V2 and V3 indicates that the V3 mac’s U-RF is the entire sheet of V2 macs; thus, the V3 mac “sees” the entire input surface (visual field). Five out of a possible

⁵ Actually, R* is the average accuracy (across the macs of the level) of SDR codes activated during recognition compared to those assigned during learning, *averaged over all frames of all sequence*. R^Ω is the average accuracy (across the macs of the level) of the SDR code activated during recognition compared to those assigned during learning averaged only over the final frames of all sequences. But in these MNIST studies, the inputs are not sequences, but just single frames. and “R_omega” is that measure averaged only over the final frames of all sequences. But in this case, the sequences only had one frame each, i.e., they were images (so R_star = R_omega).

six V2-scale features are active in V3's U-RF. This corresponds to seven active V1-scale features.

The reason for investigating smaller V0 apertures, is that as aperture size decreases, the space of possible features that can occur in the aperture decreases exponentially. In addition, the smaller the aperture, the tighter the bounds on the number of active pixels necessary for the associated V1 mac to activate (π_1^-, π_1^+) can be. These facts, combined with the strong reduction in possible features imposed by our preprocessing scheme (1-pixel-wide edge filtering), suggest that the representational bases achieving a given level of representational fidelity may be able to be much smaller and learned much more quickly for smaller apertures vs. larger apertures. We will present these comparisons on an ongoing basis.



Instance: 0 1 2 3 4 5 6 7 8 9

Figure 30: The 10 Instances of All 10 Digits with Superimposed V0 Aperture Grid

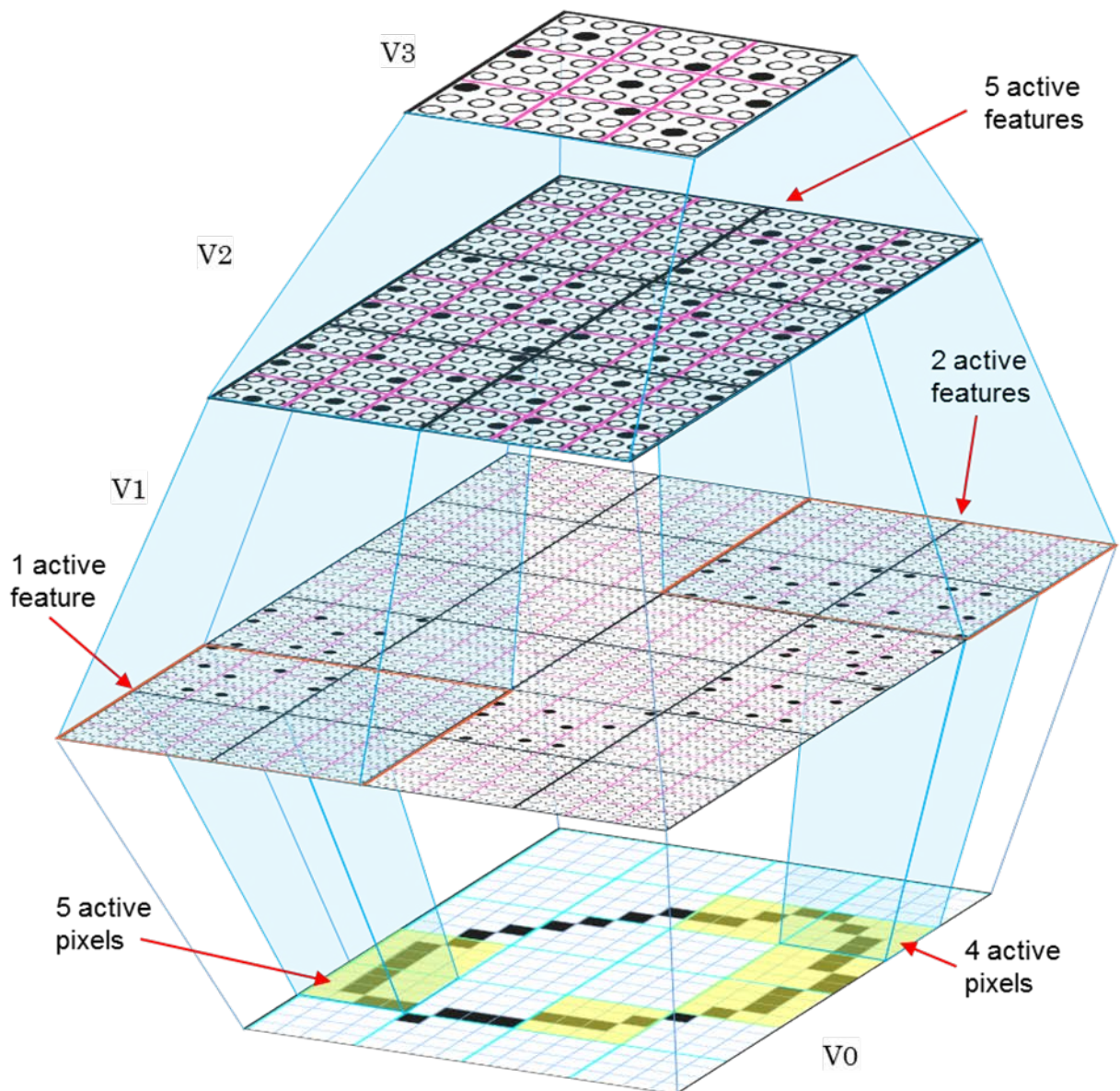
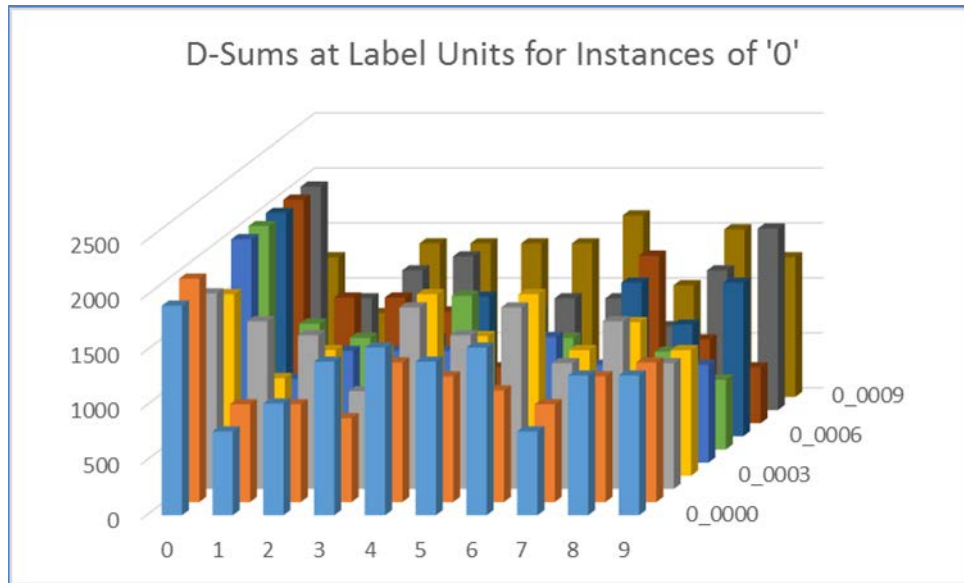


Figure 31: A Second Model Used in MNIST Digit Experiments

Table 9 gives the D-summations (arriving from the active V3 code) to all ten category, i.e., “label”, units (rows) for the test trials for the 10 instances of digit ‘0’ (columns). Category unit ‘0’ has the maximal sum for all instances except instance 3, where it is tied with category units ‘4’ and ‘6’ (orange), and instance 9, where it is beaten by most of the other category units. Figure 32 presents Table 9 in chart form. The model got 77 out of 100 test trials correct in this experiment and the results for the test trials of digits, ‘1’ to ‘9’, were similar to those for digit ‘0’: i.e., usually getting 7-8 instances correct and making a mistake on 2-3.

Table 9. D-summations at all Category Units for Test Trials for the 10 Instances of ‘0’

	Instance of Digit ‘0’									
Category	0	1	2	3	4	5	6	7	8	9
0	1905	2032	1778	1651	2032	2032	2032	2032	2032	1270
1	762	889	1524	889	762	1143	635	1143	1016	762
2	1016	889	1397	1143	1016	1016	635	1143	1270	1397
3	1397	762	889	1143	1016	889	889	1016	1397	1397
4	1524	1270	1651	1651	1016	1397	1270	508	889	1397
5	1397	1143	1397	1270	635	1016	1143	762	1016	1397
6	1524	1016	1651	1651	1143	1016	381	381	1016	1651
7	762	889	1143	1143	889	381	1397	1524	762	1016
8	1270	1143	1524	1397	889	889	1016	762	1270	1524
9	1270	1270	1143	1143	889	635	1397	508	1651	1270

**Figure 32: Chart Representation of Table 9**

We step back here to elaborate on the hierarchical nature of the recognition process. As Figure 28 shows, it is only the top-level code in the visual hierarchy that is associated with a category (label). Due to convergence / divergence of the synaptic U matrices, the top-level code is the most abstract and compact representation of the visual information, just as progressively more anterior cortical regions are the most abstract. One meaning of ‘most abstract’ is ‘most tolerant to variation across instances of a category’. In Sparsey, as in other hierarchical models, the job of dealing with variation is handled in a piecemeal, hierarchical fashion by the different macs across the different levels. During learning, each mac accrues (stores) a set of features that it experiences. During recognition, each mac finds the closest one of those stored features to its current input, activates the SDR code for that feature and sends that information out to the macs in the next higher level. In the process of doing that, each mac is handling variation in the portion of the input that it sees (i.e., its U-RF). Macs at the next higher level then perform the

same operation—i.e., finding their best-matching stored features, activating the codes for those features, and passing the information to the next level. Note that in the purely spatial case discussed here, H- and D-signals are not used, except for the D-signals from V3 to the label field. Figure 33 zooms in for a closer look at the above scenario for four of the V1 macs, macs 6, 9, 14, and 18. The key at left shows locations of these four mac’s V0 apertures. For the particular set of 100 instances used in this example (shown in Figure 30), Mac 6 experiences 49 instances in which the number of active pixels in its U-RF falls within the low and high constraints, $\pi_1^- = 4$ and $\pi_1^+ = 5$. Of these 49 instances, 27 are unique: these are shown in Figure 33. Each of these 27 unique input features is assigned a code in mac 6. In Figure 30, yellow highlight indicates the first time the feature occurs in Mac 6’s aperture, rose highlights indicate repeat occurrences. Similarly, for Macs, 9, 14, and 18. Thus, these are pictures of the learned *bases* of these four macs. Note that this experiment involved only 100 images, which is small enough so that none of the V1 macs (or any other macs) were frozen due to their afferent U-wt matrices becoming saturated. Thus, while these bases appear to cover the space of possible future 4-5-pixel inputs to their 4x4-pixel apertures with fairly high fidelity, many more features could be stored in each of the macs.

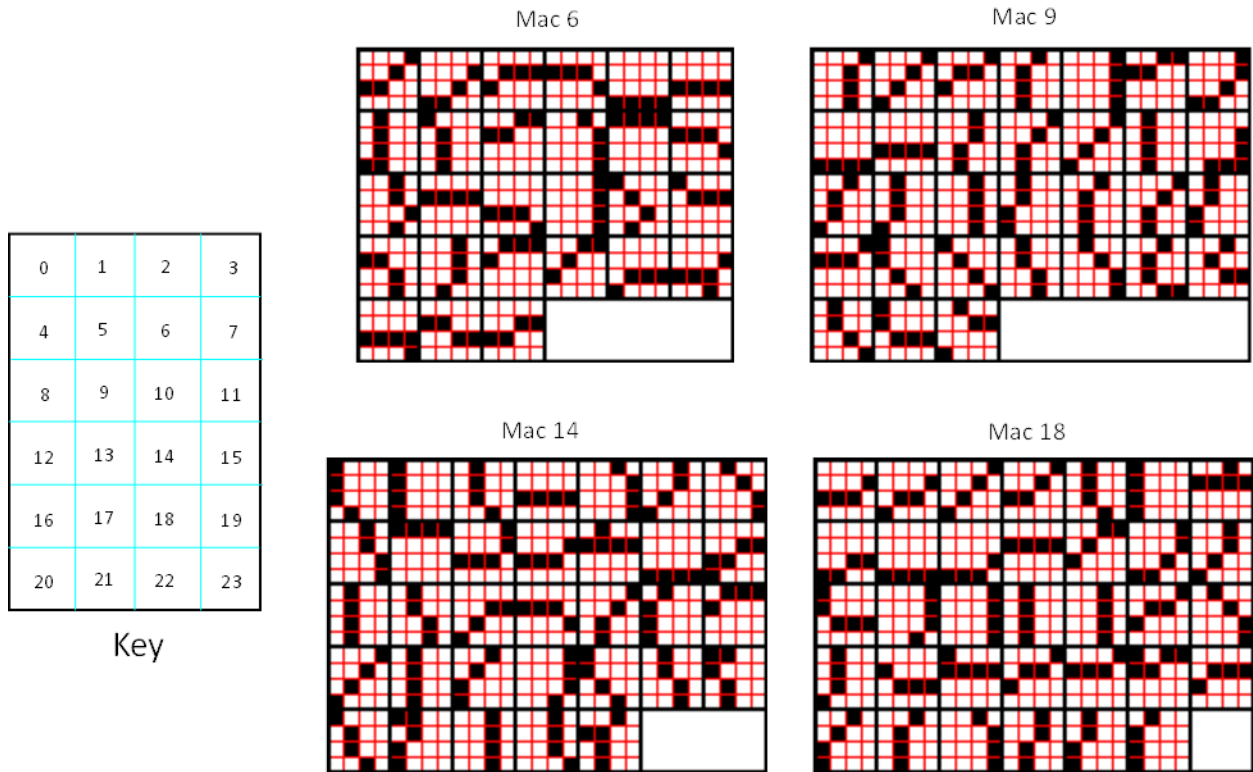


Figure 33: Learned Bases in Four V1 Macs for MNIST Experiment

We make the following points about the feature bases shown in Figure 33 and about the notion of feature bases present in Sparsey in general.

1. Every mac, at V1 and at all other levels, learns its basis from scratch. An input pattern is admitted to a mac's basis if:
 - a. It meets the constraints on the number of active features (π_1^-, π_1^+)
 - b. It is sufficiently different from any previously stored pattern so that the CSA assigns it a unique code
 - c. The total number of features stored in the mac so far is low enough that the mac has not been "frozen". Recall, that we need to freeze learning in a mac, i.e., prevent any new increases to the afferent wts to the mac, once the fraction of increased afferent wts to the mac passes a threshold.
2. A code is assigned, with full strength, to a feature upon the feature's first occurrence. Specifically, for the case of a code assigned in a V1 mac, the weights from the input units (at V0) to the mac's active units are increased to their maximum value in that single instance. However, that code does not become a permanent element of the basis unless it occurs again within a specified window of time (e.g., number of frames), which is a model parameter.⁶ This is the reason we say Sparsey requires two-trial learning, though it is important to understand that new inputs are stored (learned) with full-force upon their first occurrence (unlike models that use gradient descent, like Backpropagation, which requires numerous repetitions of an input to be learned).
3. The input patterns that become part of a mac's permanent basis reflect the statistics of the input space at the scale of that mac's aperture. At the lowest scale, in this case, a 4x4-pixel patch of input space, and given Sparsey's preprocessing (edge-filtering), we would expect the inputs experienced in any one aperture to be statistically quite similar to those of any other aperture. This is basically supported by the four bases shown in Figure 33.
4. Because of the extremely domain-neutral criteria for admitting an input to a basis (point 1 above), the actual basis elements can look rather irregular. Yes there are some very standard-looking features, e.g., exactly horizontal edges, exactly vertical edges, perfect diagonals, etc., but many seem somewhat arbitrary. However, there are strong arguments based on first principles, why bases such as these, which contain a large fraction of irregular-looking features can represent the input space just as well (perhaps even better) as a regular, engineered basis, e.g., a pre-defined set of Gabor filters. The success of compressive sensing provides further evidence for this view (Pitkow 2012).
5. Despite the fact that in describing how the mac operates during recognition, we say that it "finds the stored feature that most closely matches its input", we emphasize that that "search" process has fixed time, i.e., as the number of features stored in the mac grows, the time it takes to find the closest match remains constant. It is not any sort of *serial* search through stored items: all stored items are evaluated in parallel. Furthermore, during learning, the time

⁶ In fact, the transition from being a transient element of a mac's basis to being permanent operates at a finer granularity than 'whole features'. A weight has another parameter, *permanence*, which is increased on subsequent instances in which the weight's presynaptic and postsynaptic units are co-active if that next instance occurs within the window of time following the previous instance of pre-post coactivity. In our current experiments, weights are made permanent (permanence is set to "1") upon the second instance of pre-post co-activation.

it takes to store each new feature also remains constant as the number of stored features increases. This is true for every mac at every level for the life of any particular system. It is this characteristic—fixed-time storage and fixed-time best-match retrieval—that most strongly distinguishes Sparsey from all extant information processing algorithms.

6. Some would take point 3 above—that the bases learned in different macs at a given level will likely end up having statistically similar elements—as an argument in favor of maintaining one central representation of the basis and using it to process all the apertures of the level. This is generally done in convolutional net models (LeCun and Bengio 1995, Taylor, Fergus et al. 2010, Le, Zou et al. 2011, Lee, Grosse et al. 2011, Zeiler, Taylor et al. 2011). One reason given is such a centralized basis representation will have far fewer parameters and thus require far less training to attain any given level of fidelity to the input statistics. However, while this point is of course true (known as the ‘curse of dimensionality’ in statistics), it essentially misdirects us. The strongest empirical argument that it misdirects us is that the human brain, which is the best computational system known, patently does not use a central representation of a basis. Clearly, each individual patch of cortex (we think in terms of macs) maintains (in its weights) its own memory/basis and applies it to its own (local) inputs on an ongoing basis. There is no sense in which a central canonical representation of a V1 basis is stored in some part of the brain (or of cortex) and applied sequentially to the different macs. This would require copying/moving of far too much information. It is increasingly realized within the hardware community that moving information uses the majority of processing time and energy. The brain avoids all of this information movement because all processing is local. Sparsey shares this characteristic with biological brains and this suggests it will have perhaps exponentially lower compute times and energy usage.

Figure 34 shows the *approximate* learned bases for all six V2 macs. (Note: the switch of grid line colors from red in Figure 33 to cyan here is of no consequence.) Each V2 mac receives U-input from the four V1 macs underlying it. Thus, the V2 input apertures are 8x8-pixels. For example, V2 Mac 0 becomes active on 32 of the 100 images presented during learning, V2 Mac 1 activates for 64 of the images, etc. In this experiment, the criterion for a V2 mac to become active is that between $\pi_2^- = 1$ and $\pi_2^+ = 4$ of its afferent V1 macs is active. Recall that in this experiment, V1 macs become active if between $\pi_1^- = 4$ and $\pi_1^+ = 5$ of their afferent pixels are active. Thus, in each of the 8x8-pixel images in Figure 34, only the pixels in 4x4-pixel quadrants that meet that criteria (yellow) are actually “seen” by (and thus, encoded by) the V2 mac. The active pixels in the grayed out quadrants are not actually included in the feature being stored in the relevant V2 mac (because their intervening V1 mac does not become active). We show a few examples of this concept for some 8x8 input patterns for some of the macs. It is for this reason, that we say that Figure 34 depicts the *approximate* bases of the V2 macs.

A key point to make with respect to Figure 34 is that, with the 4x larger input aperture, i.e., 8x8 pixels instead of 4x4 pixels, the feature space for a V2 mac is exponentially larger than for a V1 mac. Nevertheless, given the edge-filtering preprocessing and the constraints on V1 activation, the space of actual inputs that a V2 mac can experience is only a tiny fraction of that overall 8x8 binary pixel space. Furthermore, we suggest that, for the purposes of recognizing higher-level objects/events, any particular feature stored in a V2 mac basis (e.g., any of those shown in Figure 34) can represent a fairly large space of similar (in terms of pixel overlap) other features reasonably well. Thus, we further suggest that bases similar to those depicted in Figure 34,

though probably somewhat larger (perhaps 100-200 features) would be able to represent future inputs with quite high fidelity. This is an empirical question which we will be investigating in detail on an ongoing basis.

In these experiments Sparsey learns, in unsupervised fashion, low-level bases, across a range of scales, in particular, at the 4x4 scale of V1 (Figure 33), at the 8x8 scale of V2 (Figure 34), and at the whole 16x24 scale of V3 (essentially, the individual inputs themselves shown in Figure 30). In addition, the basis elements (SDR codes) assigned in the single V3 mac are associated with the category-representing units in the Label field. Moreover, the basis elements (again, SDR codes) at neighboring scales are associatively linked with each other, instantiating the hierarchical composition of features (across as many scales as there are levels in the network). All of this proceeds in parallel and on the basis of (essentially) single-trial learning. There is no gradient descent, or Markov chain Monte Carlo (MCMC) sampling to estimate gradient (or gradient-like) information.

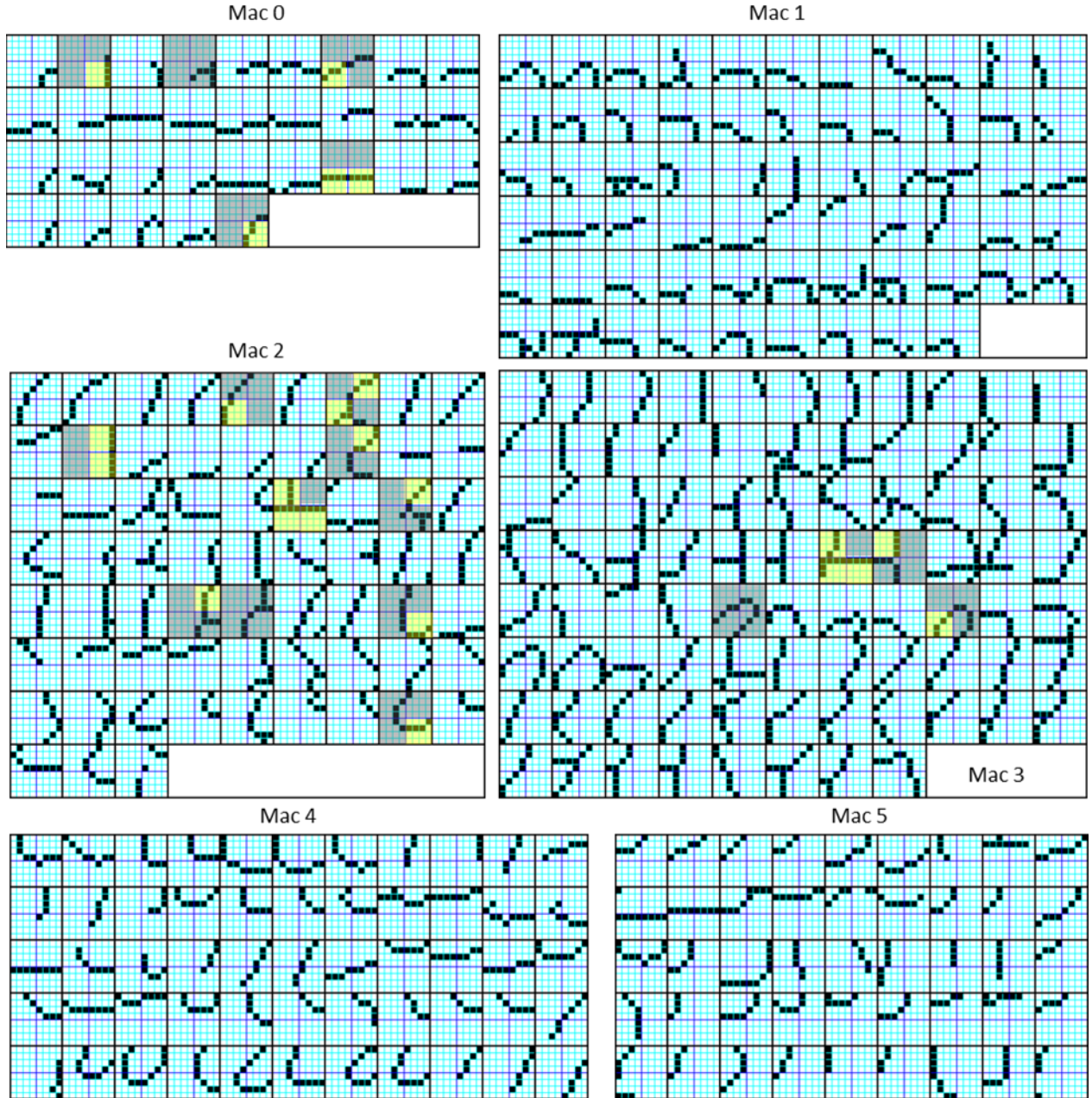


Figure 34: Approximate Bases for All Six V2 Macs

5.7.1. Re-use of Existing Knowledge in Hierarchical Networks

This study provides expanded results and discussion of the MNIST task. We describe how the hierarchical and compositional structure of naturalistic input spaces can be *efficiently* captured by (embedded in) hierarchical Sparsey networks, specifically, how the explicit hierarchical network structure, in which receptive field size and code activation duration (persistence) increase with level, will automatically capture, represent efficiently, and leverage during recognition, the hierarchical structure of natural input domains. In the case of MNIST, we are talking about

spatial hierarchical structure, but the described principles/mechanisms generalize to the spatiotemporal case as well.

Figure 35 shows the model instance used in Study 1. Its input surface (V0, a.k.a. L0) is 16x24 pixels, its V1 (L1) level is a 4x6 array of macs, each with a 4x4-pixel aperture, V2 (L2) is a 2x3 array of macs, each with a 2x2 array of V1 macs comprising its U-RF, and V3 (L3) is a single mac with all six V2 macs comprising its U-RF. The V3 mac is fully connected with the Label field, S0, which consists of 10 units representing the 10 digit classes. A sample of the D-wts from V3 to S0 are shown in gray. Magenta lines indicate increased D-wts which, in the recognition trial depicted, would yield the maximal D-summation for label unit '0'. Because some of the active V3 cells will also be included in SDR codes of instances of other digits, the other label units' D-summations will in general also be non-zero.

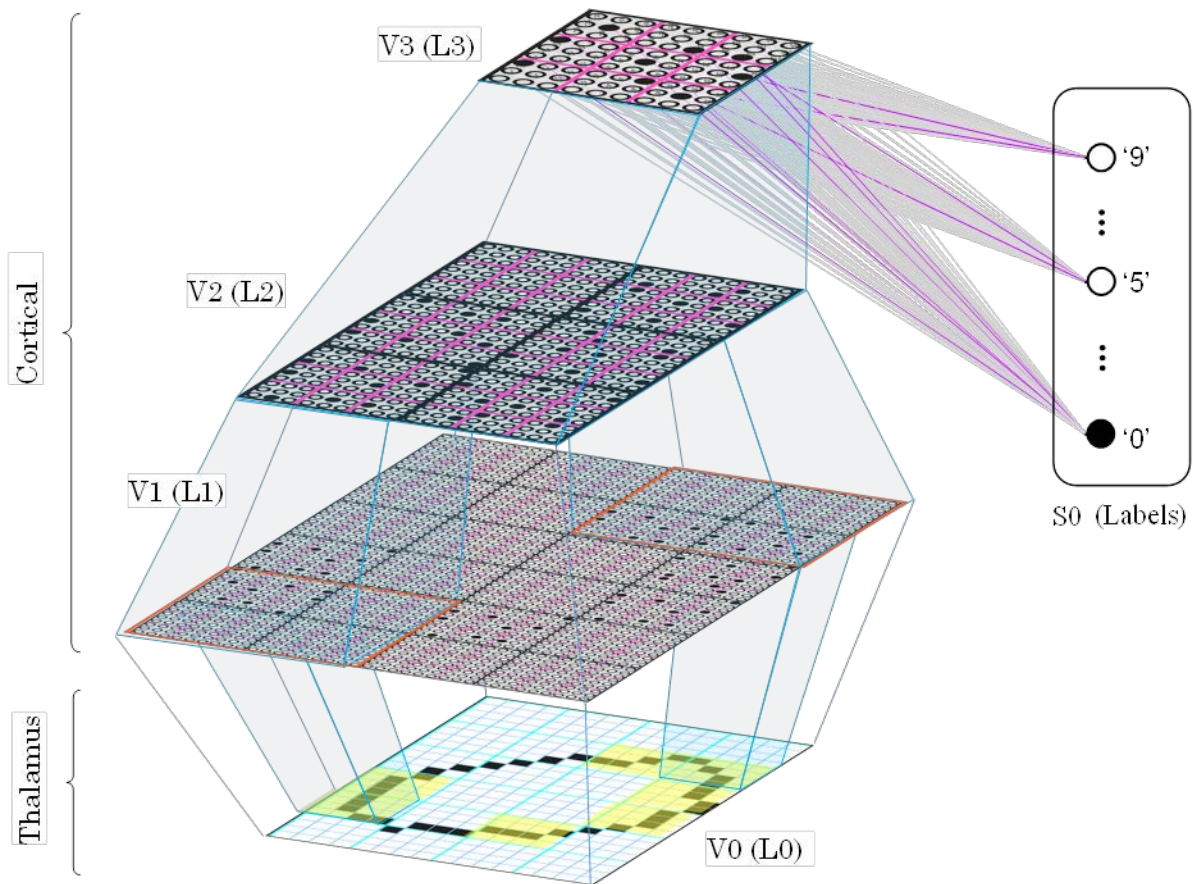
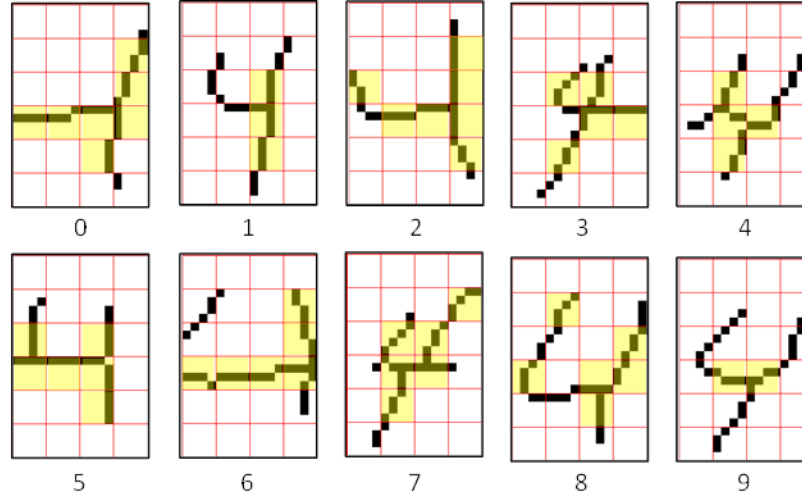


Figure 35: Sketch of Model Used in Study 1

The training set and test were the same and consisted of the 100 inputs shown in Figure 30. The 10 instances of digit '4' in the train/test set are shown in Figure 36 (Top). The 4x6 array of 4x4-pixel apertures, which are the U-RFs of the V1 macs is superimposed on each instance (red lines). Figure 36 (Bottom) shows the exact SDR codes assigned in all active macs on the single learning trial of instance 2 of digit '4'. For each active mac, we report the G value, which represents the familiarity of the U-input to the mac, and the indexes of winning units (neurons) in each of the mac's $K=16$ CMs (in this case, $K=16$ for macs at all three internal levels). In this

particular experiment, the lower and upper bounds on the number of active features (pixels) in a V1 mac's U-RF needed to cause that mac to become active are $\pi_1^- = 4$ and $\pi_1^+ = 6$. Yellow-highlighted apertures are those which satisfy those bounds. The V2 and V3 bounds are $\pi_2^- = 1$, $\pi_2^+ = 4$, $\pi_3^- = 1$, and $\pi_3^+ = 6$.



L3: [24, 0]: (active macs: 1/ 1)	
MAC 0: G: U 0.006	1 5 3 2 11 6 3 11 4 9 9 6 6 11 7 7
L2: [24, 0]: (active macs: 4/ 6)	
MAC 1: G: U 0.023	13 1 13 4 0 4 10 7 4 7 1 7 14 4 5 6
MAC 2: G: U 0.009	13 10 1 8 9 8 9 5 4 14 10 0 11 1 13 10
MAC 3: G: U 0.297	11 11 3 4 10 12 4 6 5 4 10 13 0 6 5 0
MAC 5: G: U 0.003	11 8 6 0 5 0 13 10 9 9 8 7 1 7 8 10
L1: [24, 0]: (active macs: 7/ 24)	
MAC 7: G: U 0.289	12 8 12 5 13 14 14 8 10 0 13 12 0 10 10 4
MAC 8: G: U 0.000	5 10 1 12 12 4 0 13 5 0 13 14 12 4 14 14
MAC 11: G: U 1.000	2 13 12 13 2 1 1 13 4 5 13 7 14 0 5 2
MAC 13: G: U 0.590	13 12 0 3 1 6 3 11 5 7 13 9 8 11 5 5
MAC 14: G: U 1.000	10 9 0 7 1 4 9 8 5 8 2 8 11 12 0 14
MAC 15: G: U 1.000	8 13 5 2 13 6 10 9 1 8 12 1 4 0 5 13
MAC 19: G: U 0.063	10 14 7 2 6 3 3 9 5 14 7 5 9 5 4 8

Figure 36: Ten Digit ‘4’ Instances in Train/Test Set and Detailed SDR Codes for All Active Macs for Instance 2 Learning Trial

The information shown in Figure 36 (Bottom) is for the learning presentation of instance 2 (the 3rd instance) of digit ‘4’. Up to this point, the model will have been presented with three instances each of digits, ‘0’, ‘1’, ‘2’, and ‘3’, and two instances each of digits ‘4’ through ‘9’. Notice that $G=1$ for three L1 macs, M_{11}^1 , M_{14}^1 , and M_{15}^1 . This means that on this, the 25th learning trial, the patterns in these mac’s apertures are being judged as completely (100%) familiar. In other words, on one or more of those 24 prior training trials, the same exact pixel pattern has occurred in each of these apertures. Figure 37 shows these two prior instances, instance 0 of digit ‘2’ (bottom left) and instance 0 of digit ‘4’ (bottom right). Rows highlighted in same color indicate codes for which the input pattern (to the relevant mac’s aperture) was identical. The identical feature patterns between instances are highlighted in color-keyed fashion and linked with the corresponding SDR codes.

In M_{11}^1 , the same vertical bar that appears in the learning trial for instance 2 of digit ‘4’ (bottom center) also appeared in the learning trial for instance 0 of digit ‘2’ [highlighted in blue, and blue paths link to the specific SDR codes activated (also highlighted in blue) in those instances.] Note that even though the input in M_{11}^1 ’s aperture is identical in these two instances, the SDR codes differ by two cells (red underlines). This is due to CSA’s probabilistic nature: even when $G=1$ (perfect familiarity), the *maximally-implicated* cell (i.e., the cell with the highest V value in its CM) sometimes loses. Nevertheless, the two SDR codes still have 14 out of 16 cells in common. Also, notice that the pattern in M_{11}^1 ’s aperture in instance 0 of ‘4’ has only one pixel in common with the pattern in the other two instances. Due to Sparsey’s SISC property, this results in a very different SDR code becoming active in M_{11}^1 (dashed blue arrow).

Both the horizontal bar in M_{14}^1 ’s aperture (green) and the vertical bar in M_{15}^1 ’s aperture (green), which occur in the learning trial for instance 2 of digit ‘4’ also appeared in the learning trial for instance 0 of digit ‘4’. Consequently, the two M_{14}^1 codes have 15 out of 16 cells in common. The two M_{15}^1 codes only have 11 out of 16 cells in common. Again, this is because parameters allow non-maximally-implicated cells to win with some probability. Furthermore, the precise details of the specific history of other patterns that have occurred in M_{15}^1 and the codes to which they have been assigned influence the win probability distributions in the CMs. We could have used different parameters for this experiment which would have resulted in *exactly* the same codes becoming active in M_{14}^1 and M_{15}^1 in instance 2 of digit ‘4’.

The importance of Figure 37 is that it shows the re-usage of previously learned lower-level (specifically, L1) features in new higher-level (specifically, L2) concepts. The horizontal bar feature that is stored in M_{14}^1 during presentation of instance 0 of ‘4’ is reactivated again during the learning trial of instance 2 of ‘4’. By “reactivation” of the feature, we mean that the SDR code of that feature is reactivated. To be precise, that code is almost precisely reactivated, i.e., 15 out of 16 of its cells are reactivated. As noted above, we can adjust parameters to make the probability of perfectly reinstating the entire code as close as we want to unity. So, let’s assume for a moment that:

- the SDR code activated in M_{11}^1 in instance 2 of ‘4’ is identical to the code activated in M_{11}^1 in instance 0 of ‘2’, and that
- the SDR codes activated in M_{14}^1 and M_{15}^1 are identical the codes activated in those macs during instance 0 of ‘4’.

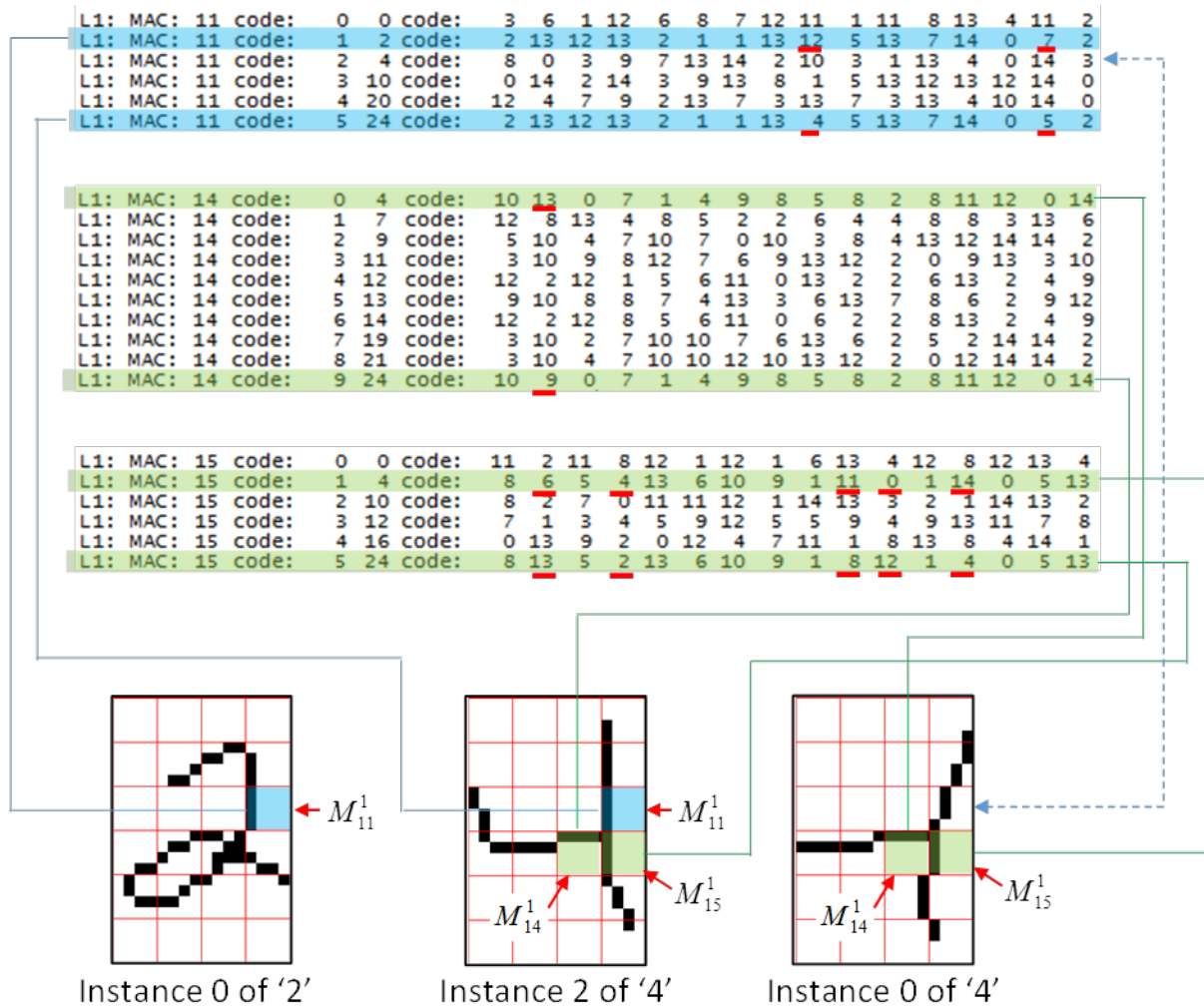


Figure 37: The Two Prior Learning Trials with Identical Inputs in Some Apertures and Codes in the Associated L1 Macs

Under these assumptions, there is *no* new learning between L0 and these three L1 macs on the learning trial of instance 2 of '4'. However, there will still be learning from these L1 macs to the L2 macs to which they project. For example, M_3^2 's U-RF consists of L1 macs, M_{10}^1 , M_{11}^1 , M_{14}^1 , and M_{15}^1 . Even though, in the present instance (the learning trial of instance 2 of '4'), all three active macs in M_3^2 's U-RF have previously learned codes active (we will refer to previously learned codes as *old* codes), those three codes have in fact never occurred together. Thus, the overall input to M_3^2 is novel and a new code is activated in M_3^2 . Since M_3^2 's code is new, there will in general be new learning from all of M_3^2 's afferent L1 macs, even from those in which old codes are active.

Figure 38 compares the conditions during the learning trials of instance 0 (panel a) and instance 2 (panel b) of digit '4'. Figure 38a (Left) shows the pixel patterns active in the L0 apertures "seen by" the L1 and L2 macs on the learning trial of instance 0 of digit '4'. The seven active L1 macs are highlighted in yellow. The blue-outlined rectangular prism shows the four L1

macs comprising the U-RF of L2 mac, M_3^2 , three of which, M_{11}^1 , M_{14}^1 , and M_{15}^1 , are active. Thus, we say that M_3^2 has three active features in its U-RF. Consequently, M_3^2 is active. The feature hierarchy (Figure 38a, Right) shows a *compositional* view of the features at all three internal scales (L1, L2, and L3).⁷ We might reasonably term the concept active in M_3^2 as a “T junction”: it’s not a perfect “T” but pretty close. It is composed of the three L1-scale features shown and includes their specific spatial arrangement. The feature hierarchy also shows that the concept active at the top level (L3) is composed of the four L2 features shown, which are active in L2 macs, M_1^2 , M_2^2 , M_3^2 , and M_5^2 , and also includes their specific spatial arrangement.

Now consider Figure 38b. Figure 38b (Left) shows the pixel patterns active in the apertures seen by L1 and L2 macs on the learning trial of instance 2 of digit ‘4’. Overall, instance 2 of digit ‘4’ is quite different from instance 0. However, the portions of the image falling in the apertures of L1 macs, M_{14}^1 and M_{15}^1 , are the same as in instance 0 (highlighted with green glow in the feature hierarchies of panels a and b). The third L1 feature active in M_3^2 ’s U-RF, i.e., the feature falling in M_{11}^1 ’s aperture, differs greatly between the two instances. Consequently, the *overall* U-input to M_3^2 differs significantly between the two instances. The two different overall pixel patterns (features) present in M_3^2 ’s U-RF are highlighted with orange glow in panels a and b.

Consequently, the SDR code assigned in M_3^2 in instance 2 of digit ‘4’ will differ significantly (have low intersection with) the code assigned in M_3^2 in instance 0. This can be seen by comparing the SDR code active in M_3^2 in instance 2 of ‘4’ (Figure 36 (bottom), violet highlight) with the one active in instance 0 of ‘4’ (Figure 39, violet highlight): in fact, the two codes have six out of 16 cells in common. Thus, a new feature/concept (a new instance of a “T Junction”) will be stored in M_3^2 and it will include (i.e., be synaptically linked with) two old features (in M_{14}^1 and M_{15}^1) and one new feature (in M_{11}^1). *This demonstrates the re-use of old knowledge in the construction of new higher-level concepts.*

In addition, one can see that the patterns present in the other three active L2 macs differ greatly between panels a and b. Thus, when instance 2 of digit ‘4’ is presented, the U-inputs from L2 to L3 will be very different from what they were when instance 0 was presented. Thus, the SDR assigned at L3, i.e., the representation of instance 2 of ‘4’ (as a whole) will have low intersection with the SDR code assigned to instance 0 of ‘4’. These two codes are highlighted with aqua glow in Figure 36 and Figure 39: they have only one cell in common. Consequently, there would be a lot of new learning between the four active L2 macs and the active L3 mac, all of which have new codes active.

⁷ We refer to the top-level concept here, i.e., the instance of digit ‘4’, as a “feature”: operationally, this concept is just an input pattern to a mac and is handled in the same way as an input to a mac at any other level.

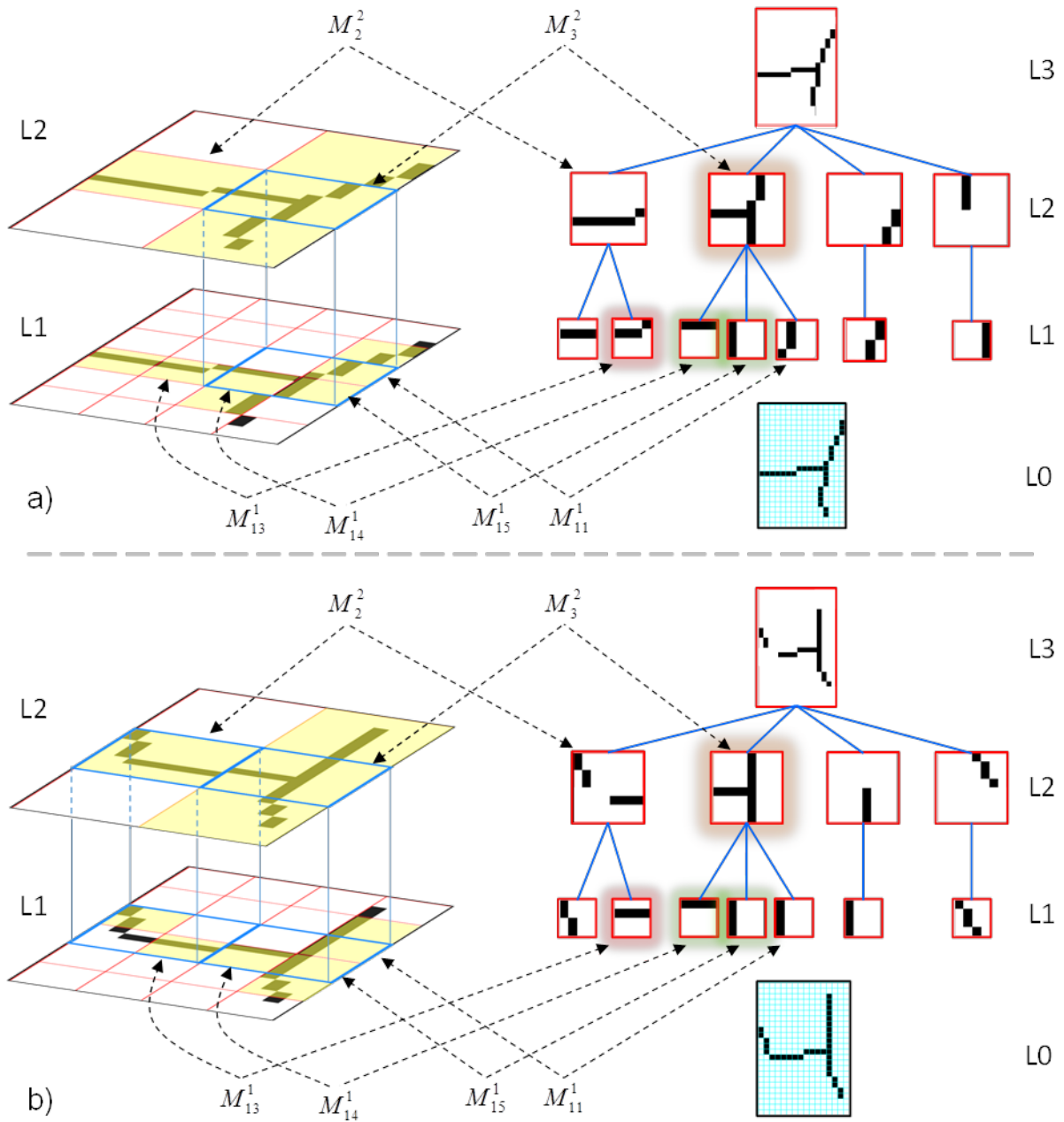


Figure 38: Detailed Pixel Patterns Present for Two Instances and Compositional Views of Featural Hierarchies Comprising Overall Concepts

L3: [2, 0]: (active macs: 1/ 1)	
MAC 0: G: U 0.000	1 1 10 6 8 1 11 5 1 6 6 11 4 9 4 5
L2: [2, 0]: (active macs: 4/ 6)	
MAC 1: G: U 0.000	2 7 2 9 3 10 5 11 14 11 11 7 2 12 7 12
MAC 2: G: U 0.000	6 2 8 5 11 5 7 6 2 4 9 10 9 5 12 14
MAC 3: G: U 0.000	13 2 11 13 4 6 6 9 4 3 9 7 14 1 9 0
MAC 4: G: U 0.000	1 14 8 1 9 13 10 12 7 7 1 3 0 12 6 14
L1: [2, 0]: (active macs: 5/ 24)	
MAC 6: G: U 0.063	5 13 1 9 6 2 0 2 3 0 8 1 10 7 8 9
MAC 11: G: U 0.000	2 13 12 13 2 1 1 13 12 5 13 7 14 0 7 2
MAC 13: G: U 0.000	13 7 9 11 11 12 10 0 14 0 12 9 12 13 10 3
MAC 16: G: U 0.250	11 14 7 13 2 8 9 7 3 11 12 0 14 4 2 11
MAC 17: G: U 0.063	2 4 14 3 5 5 0 12 10 4 4 12 11 4 10 9

L3: [4, 0]: (active macs: 1/ 1)	
MAC 0: G: U 0.000	6 1 10 4 10 1 1 2 6 2 10 6 10 1 11 2
L2: [4, 0]: (active macs: 4/ 6)	
MAC 1: G: U 0.011	8 2 10 11 3 0 0 10 11 14 12 5 11 7 0 1
MAC 2: G: U 0.009	10 7 0 9 2 6 7 1 4 13 6 8 9 8 7 6
MAC 3: G: U 0.000	11 5 5 4 8 5 4 12 14 4 2 4 5 6 7 0
MAC 5: G: U 0.009	0 9 13 12 1 2 7 13 14 9 2 13 7 14 2 8
L1: [4, 0]: (active macs: 7/ 24)	
MAC 7: G: U 0.250	1 3 1 10 10 10 5 4 10 12 9 2 5 3 14 10
MAC 11: G: U 0.063	8 0 3 9 7 13 14 2 10 3 1 13 4 0 14 3
MAC 12: G: U 0.063	7 5 14 10 4 6 4 2 2 3 8 13 9 11 14 1
MAC 13: G: U 0.563	13 13 0 11 11 12 3 6 5 3 12 14 8 13 14 9
MAC 14: G: U 0.000	10 13 0 7 1 4 9 8 5 8 2 8 11 12 0 14
MAC 15: G: U 0.000	8 6 5 4 13 6 10 9 1 11 0 1 14 0 5 13
MAC 18: G: U 0.250	3 9 10 13 12 0 0 10 1 3 2 11 14 0 8 13

Figure 39: Full Hierarchical Codes (Memory Traces) for Instance 0 of Digit ‘2’ (top) and Instance 0 of Digit ‘4’ (bottom)

Figure 40a contains the original feature hierarchy (tree) from Figure 38b. Figure 40b shows an alternate version of instance 2 of ‘4’ in which we simply made the input pattern in aperture 11 identical to its input pattern in instance 0 of ‘4’. Clearly, this is about as likely as any other instance of ‘4’ seen in the MNIST database. But, in this case, which we will call instance 2’ of ‘4’, not only do the U-RFs for M_{11}^1 , M_{14}^1 , and M_{15}^1 , have the same patterns as in instance 0 of ‘4’, so does the U-RF for the L2 mac, M_3^2 . Thus the codes that become active in macs, M_{11}^1 , M_{14}^1 , and M_{15}^1 , and M_3^2 in instance 2’ of ‘4’ would (with likelihood that we can make arbitrarily high depending on parameters) be identical to those activated in these macs in instance 0 of ‘4’. Consequently, there would be no new learning in the U-matrices represented by the green lines in panel b. Thus, this would be case in which a *whole* higher-level feature, which is itself a hierarchical sub-tree of SDR codes linked by large numbers of previously increased synapses, i.e., “old knowledge”, is *re-used* in the representation of a new overall input, instance 2’ of ‘4’.

Figure 40c shows another alternative version of instance 2 of ‘4’, instance 2’’ of ‘4’, in which the inputs to apertures 12 and 13 are also the same as in instance 0 of ‘4’. This results in the reactivation of known SDR codes in both M_2^2 and M_3^2 , constituting a greater degree of re-use of old knowledge than in panel a. Again, green lines represent U-matrices in which no new learning will occur. Figure 40d shows another, further modified version of instance 2 of ‘4’, which is in fact, identical to instance 0 of ‘4’, and which we therefore call instance 0’ of ‘4’, except that we’ve added some additional edge segments, in the apertures of M_3^1 , M_4^1 , and M_8^1 ,

which are not noise but also do not have enough pixels to cause those macs to become active. We've also added some salt and pepper noise. However, because neither the noise nor the small edge segments cause any macs to activate that were not active in instance 0 of '4' or any new input pattern in any of the L1 macs that were active in instance 0 of '4', the overall input causes the same codes to become active at all macs, including the L3 mac, as in instance 0 of '4'. Thus, the model recognizes this novel input as identical to instance 0 of '4'. In this case, the entire old memory trace is reactivated, affording no opportunity for new learning. We could call this "complete re-usage of old knowledge"; however, it is better referred to as recognition.

This principle applies recursively through all levels of the hierarchy. Sparsey determines, on a piecemeal basis across the different macs throughout the hierarchy, which fragments of its overall input are novel (and their degrees of novelty, though this nuance was suppressed in this example) and which are familiar. It re-uses (reactivates) old codes in macs whose inputs are familiar and automatically does so as far up the hierarchy as appropriate. It automatically integrates (synaptically links) old codes with new codes as appropriate. For example, Figure 40c shows another alternative version of instance 2 of '4', instance 2'' of '4', in which the inputs to apertures 12 and 13 are also the same as in instance 0 of '4', which results in M_2^2 having the same input pattern as in instance 0 of '4'. Thus, even more old knowledge is re-used in this case.

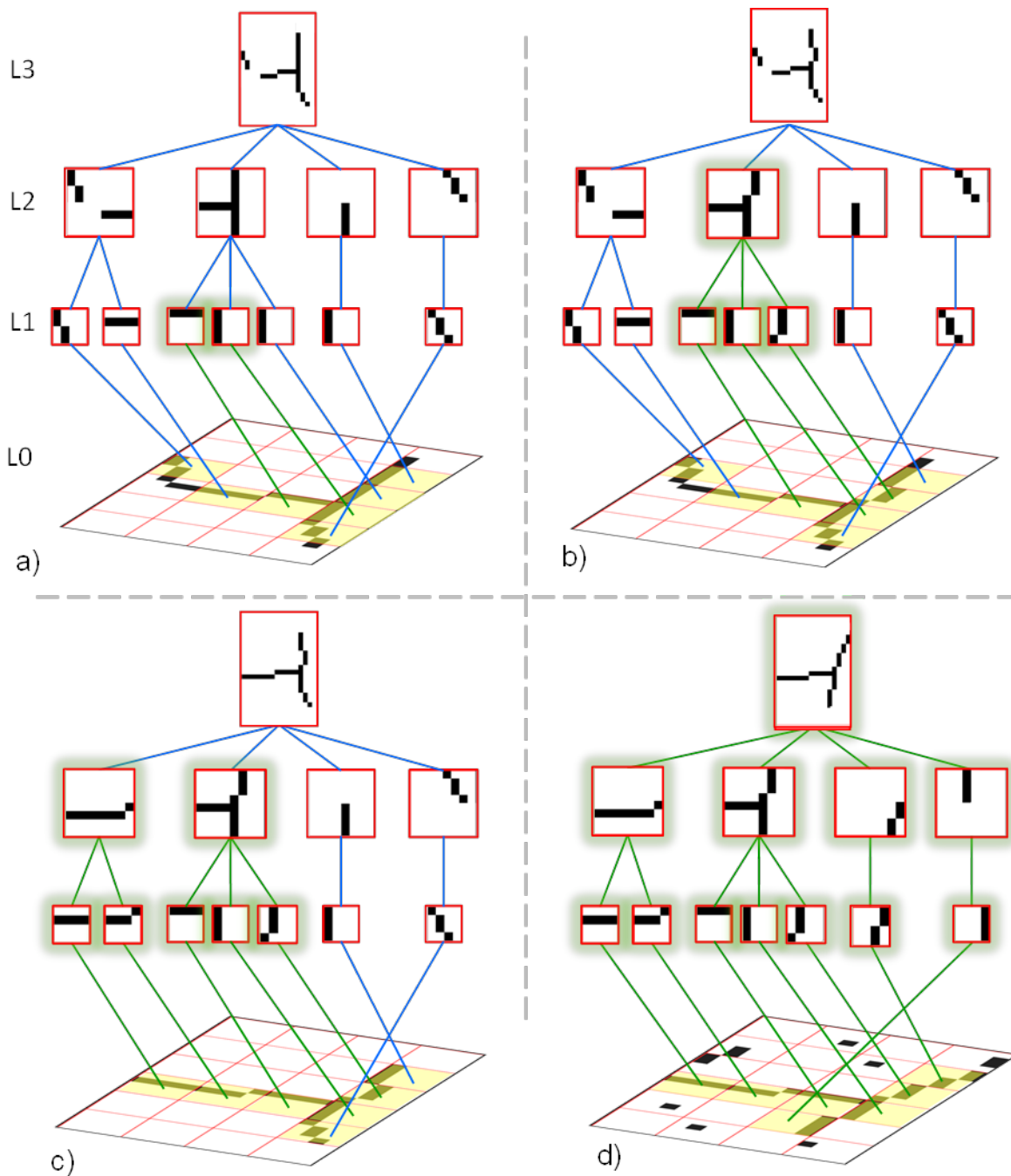


Figure 40: A Study of Invariances Across Three Scales

The principle we described in preceding figures—i.e., re-use of whole neural subtrees spanning multiple cortical regions—provides one instantiation of what has been referred to as “schema-based learning”. This is learning in which new items of knowledge are integrated with existing knowledge, i.e., into existing *schemas*. Note that there is also another principle/mechanism by which Sparsey instantiates schema-based learning. Specifically, it maps more similar inputs to more highly intersecting SDR codes (cell assemblies). Thus, when novel inputs contain sub-portions that are familiar, those sub-portions cause the number of neurons activated in response

to the novel input that were also activated in response to the prior similar sub-portion to increase. The greater the number of currently active neurons that were also active in the prior instance, the less new learning that occurs (and that needs to occur) in the current instance. This principle of schema-based learning, and more generally of hierarchical organization of representations, is possible even in a completely flat representation (cf. the recurrent neural net models of Elman, Jordan, Plate, and others). Evidence for this type of schema-based learning in hippocampus has recently been reported (McKenzie, Frank et al. 2014, O'Neill and Csicsvari 2014).

5.8 Large-Scale Episodic Memory Capacity Study

The results of this study are reported in Table 10. Also see [this page](#) on Neurithmic's web site for animation and further discussion. The table shows the R^* and R^Ω accuracies separately for all five snippets and for all seven internal levels, V1-PFC. The 8-level model used is shown (on a particular frame during processing Snippet 1) in Figure 41 (Table levels V1-PFC correspond to L1-L7 in the figure). Neurons are not shown here. We only show active macs (red border and shaded). A tiny fraction of the U, H, and D signals that occur during the trace are shown in blue, green, and magenta, respectively. The model is much larger than those previously used, having 3,261 macs and ~75 million weights. The five snippets ranged in length from 36 to 76 frames and the input surface was 64x64 pixels. The primary result shown by the table is that the model's recognition traces are essentially perfect from level PIT and up, and very good from V2 up. We emphasize that even the shortest snippet (36 frames) involves thousands of precisely coordinated mac activations across all levels (and thus tens of thousands of cell activations). And, learning was single-trial. We will continue to test how many such snippets can be stored in the model throughout the remainder of the research.

Table 10. Results of 64x64 Snippet Episodic Recognition Memory Study

Snippet	1		2		3		4		5		Ave. All Snips	
	R^*	$ROmega$	R^*	$ROmega$	R^*	$ROmega$	R^*	$ROmega$	R^*	$ROmega$	R^*	$ROmega$
PFC	0.98	1.00	1.00	1.00	1.00	1.00	0.99	1.00	1.00	1.00	0.99	1.00
P/E-rhinal / PFC	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
AIT	0.98	1.00	0.99	0.99	0.96	0.93	0.98	1.00	0.99	1.00	0.98	0.98
PIT	0.97	0.99	0.97	0.96	0.94	0.96	0.97	0.98	0.95	0.96	0.96	0.97
V4	0.94	0.96	0.91	0.87	0.89	0.96	0.91	0.90	0.86	0.88	0.90	0.91
V2	0.86	0.94	0.85	0.84	0.84	0.93	0.87	0.82	0.84	0.84	0.85	0.87
V1	0.59	0.69	0.55	0.60	0.59	0.77	0.61	0.53	0.50	0.51	0.57	0.62

The table also shows another interesting phenomenon. Specifically, accuracy is poor at V1 and generally increases with level. This seems counterintuitive because one might think that if codes begin to go wrong at the lowest cortical level, then they surely must get worse, i.e., compound, at higher levels. However, this clearly does not happen. Essentially, Sparsey's macs implement a kind of associative clean-up memory with every new code chosen. We discussed this important phenomenon in the previous report and will continue analyzing it in subsequent work.

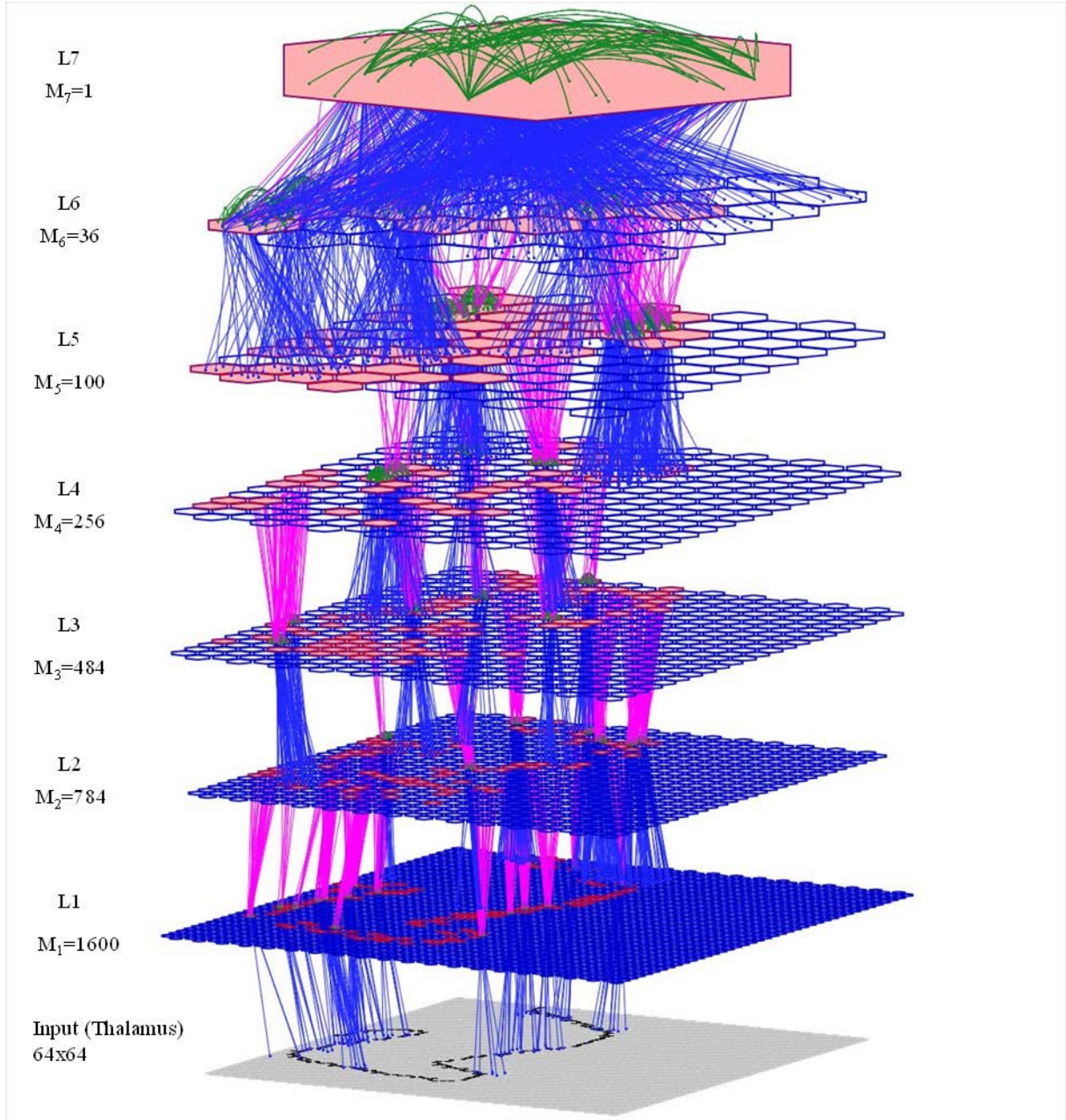


Figure 41: The 8-level Network Used in Large-Scale Episodic Memory Study

5.9 Effect of π -bounds \times U-RF Interaction on Capacity/Accuracy

The goal of this study was to verify basic parameter dependencies affecting how many snippets could be stored and with what overall recognition accuracy in larger scale models than we have used in the past. Figure 42 shows three of the 32x32 pixel snippets used and Figure 43 shows the 7-level network tested. The U-RF and π -bound parameters are constant across all five experiments, as are the numbers of macs per level (M_i , shown at left in Figure 43) and the total number of macs, which was 476. Table 11 shows some of the relevant parameters and accuracy

measures for a small sample of the configurations tested. We chose this particular sample to illustrate some of the more important parameter dependencies.

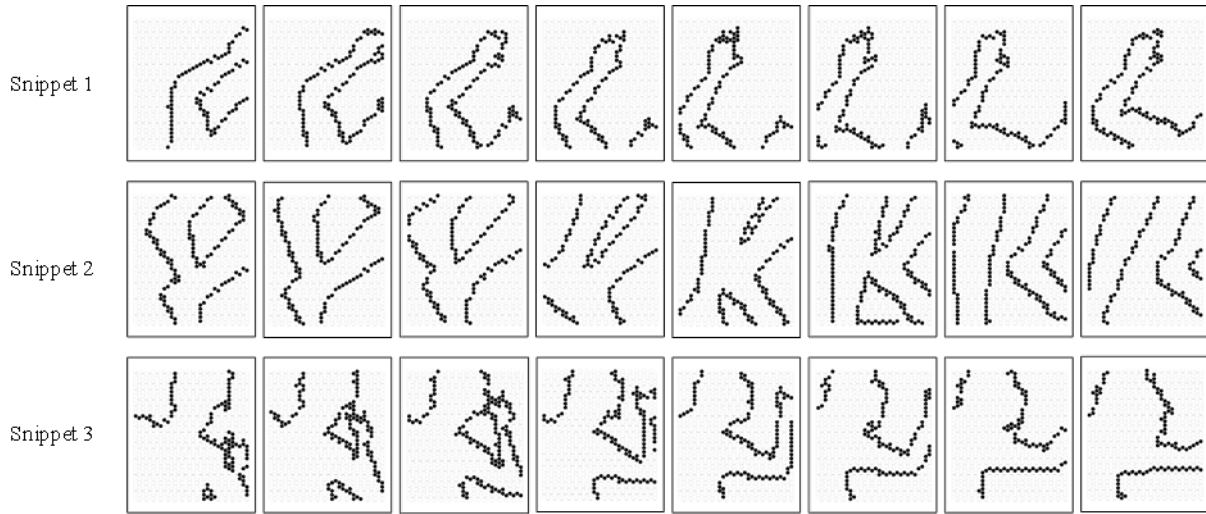


Figure 42: Three of the 8-Frame 32x32 Snippets Used in Section 5.9 Studies

The primary result of Exp. 1 is that the model attained very high recognition accuracy, $R^* = 0.93$, $R^\Omega = 0.95$, and classification accuracy when the train/test set consisted of just 15 snippets (a total of 120 frames). This, despite the use of quite small Q and K values, e.g., 7 and 7, compared to their proposed analogous values in the real brain, e.g., 70-100 and ~ 30 , respectively, and thus to the values that we anticipate in full-scale instances of the model. We emphasize that recapitulation of any one of the snippets involves the precisely timed coordination (which is learned during the single learning trial of a each snippet) of hundreds of mac activations and thousands of cell activations over the snippet's 8 frames. Figure 43 shows a tiny sample of U-RFs of several macs (blue line sprays) to suggest how information about any given local patch of pixels filters up through the levels. Of course, tens of thousands of top-down (D) and horizontal (H) signals are also being sent and processed (i.e., combined with the U-signals) during the recapitulation of a snippet.

Despite the small Q and K values used in Exp. 1, the model has a large number—approximately 7.5 million—of (binary) weights. Thus, the total information stored in bits/synapse is low. We therefore ran Exp. 2 in which twice as many snippets were presented to the model to see how the recognition accuracy held up. As one can see, recognition accuracy fell sharply to $R^* = 0.67$, $R^\Omega = 0.68$. In Exp. 3, we increased the Q and K values for the first two internal levels, L1 and L2, to 9 and 9, to see how much recognition accuracy could recover to the high levels of Exp. 1. Unfortunately, they only recovered to $R^* = 0.75$, $R^\Omega = 0.76$, despite the fact that the total number of weights rose to over 16 million. In fact, in Exp. 4, increasing L2's Q and K values to 11 and 11, increases the total weights to ~ 23.5 million, but only modestly increases recognition accuracy relative to Exp. 3. We rounded out this study with Exp. 5, in which we increased train/test set size to 45 snippets, and again saw a sharp decrease in recognition accuracy to $R^* = 0.53$, $R^\Omega = 0.52$. Thus, simply increasing Q and K , and indirectly, the number of weights, especially if done without coordinated changes in other parameters, is not a trajectory

through parameter space that allows bits/synapse to remain constant or increase while maintaining high recognition accuracy. Nevertheless, based on many other experiments we have performed, we believe that information storage capacity can be greatly increased beyond the results seen in this study by tightly coordinated setting, across all model levels, of other parameters, most notably the U-RF sizes and π -bounds, but also the weight saturation thresholds of the various afferent U, H, and D matrices.

Table 11. Recognition Accuracy Results for this Study

Exp	Lev	M	Q	K	Ave. U-RF	Ave π	Num. Snips.	Num. Weights	Class. Acc.	R^*	R^Ω	Ave	
												R^*	R^Ω
1	6	1	6	6	9	[1,9]	15	7401474	14/15	1.0	1.0	0.93	0.95
	5	9	6	6	5	[2,3]				0.99	0.99		
	4	25	7	7	6	[2,3]				0.95	0.97		
	3	64	7	7	5	[1,2]				0.91	0.93		
	2	121	7	7	5	[1,1]				0.89	0.93		
	1	256	7	7	14	[4,4]				0.84	0.87		
2	6	1	6	6	9	[1,9]	30	7401474	16/30	0.65	0.65	0.67	0.68
	5	9	6	6	5	[2,3]				0.67	0.68		
	4	25	7	7	6	[2,3]				0.65	0.69		
	3	64	7	7	5	[1,2]				0.67	0.69		
	2	121	7	7	5	[1,1]				0.70	0.71		
	1	256	7	7	14	[4,4]				0.68	0.67		
3	6	1	6	6	9	[1,9]	30	16284560	18/30	0.72	0.72	0.75	0.76
	5	9	6	6	5	[2,3]				0.75	0.74		
	4	25	7	7	6	[2,3]				0.72	0.73		
	3	64	7	7	5	[1,2]				0.78	0.78		
	2	121	9	9	5	[1,1]				0.81	0.85		
	1	256	9	9	14	[4,4]				0.74	0.77		
4	6	1	6	6	9	[1,9]	30	23468718	20/30	0.77	0.77	0.78	0.8
	5	9	6	6	5	[2,3]				0.76	0.76		
	4	25	7	7	6	[2,3]				0.74	0.74		
	3	64	7	7	5	[1,2]				0.83	0.82		
	2	121	11	11	5	[1,1]				0.84	0.9		
	1	256	9	9	14	[4,4]				0.75	0.79		
5	6	1	6	6	9	[1,9]	45	23468718	12/45	0.44	0.44	0.53	0.52
	5	9	6	6	5	[2,3]				0.49	0.48		
	4	25	7	7	6	[2,3]				0.47	0.45		
	3	64	7	7	5	[1,2]				0.56	0.52		
	2	121	11	11	5	[1,1]				0.63	0.63		
	1	256	9	9	14	[4,4]				0.61	0.59		

Key: “Ave. U-RF” is the ave. # afferent features (pixels for L1 macs, subjacent macs for L2 macs and higher) for macs of a given level. “Ave π ”: the π range consisting of the ave. π^- and ave. π^+ over all macs of a level. Red text indicates values changed from prior experiment.

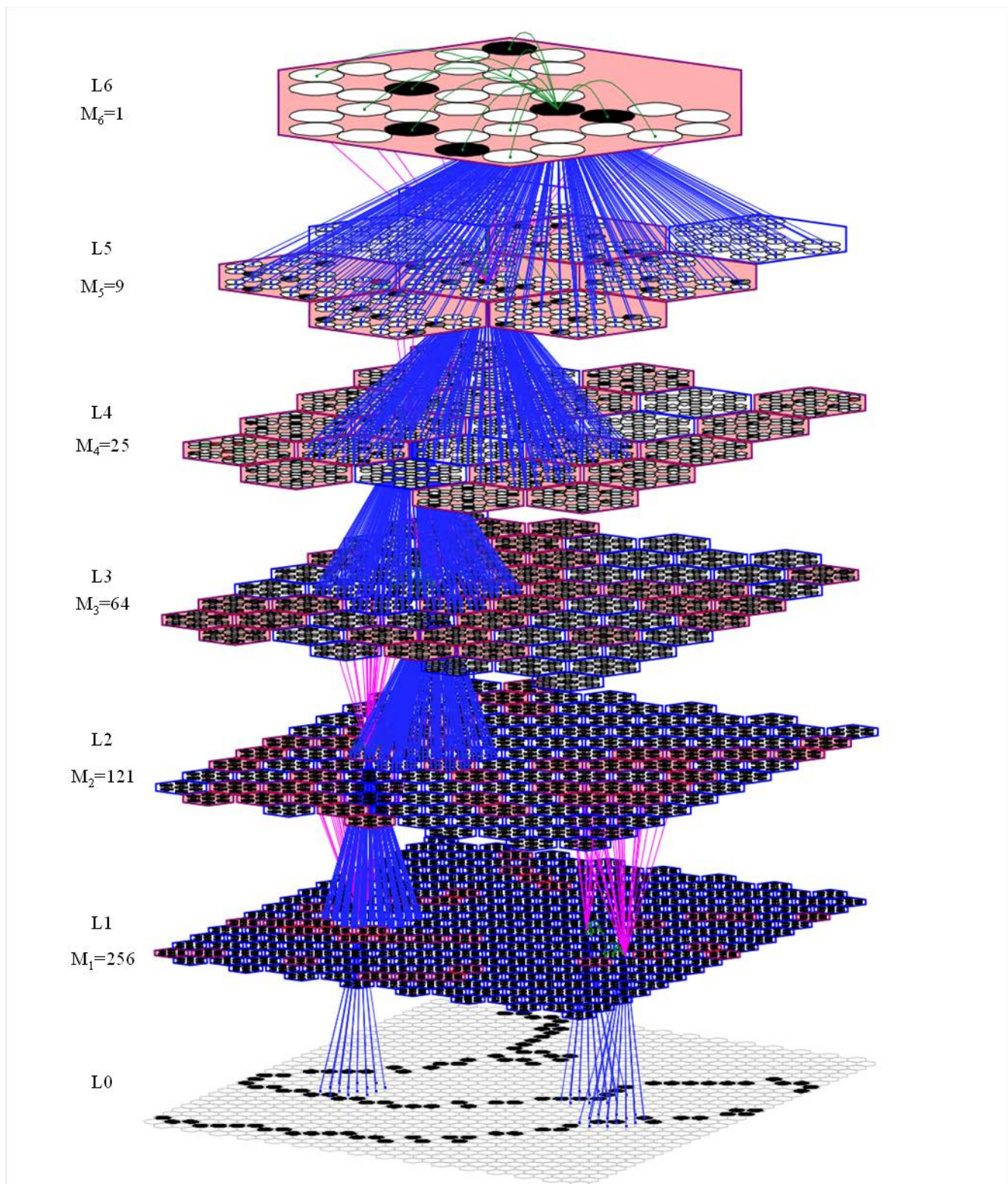


Figure 43: Snapshot of 7-Level Model Used in Section 5.9 Studies

We tried many combinations of U-RFs and π -bounds at the different levels and for models with different numbers of levels. Table 12 shows some of the relevant parameters and accuracy measures for a small sample of the configurations tested. Exp. 1 in this table is the same as in Table 11 and is given as a baseline to which to compare the other experiments in this study. Again, the model in Exp. 1 shows high recognition accuracy for the 15-snippet train/test set. Furthermore, separate analyses show that the fraction of input features represented at all levels including the top level (L6) remains high. We point out that the L1 mac π bounds are as tight as possible. The U-RF of almost all of the 256 L1 macs has 14 pixels, with π bounds,

$[\pi^- = 4, \pi^+ = 4]$; thus, exactly 4 out of 14 pixels had to be active for the L1 mac to become active. Recall that in earlier reports we have explained that in general, to minimize interference (cross-talk) between stored codes, the range of input pattern sizes (numbers of active features) that cause a mac to activate (and thus, which are stored) should also be minimized, ideally to zero. This allows all input patterns stored in a mac to be normalized on an equal (unbiased) footing.

To verify the superiority of tighter π bounds, in Exp. 2, we relaxed the L1 π bounds to

$[\pi^- = 3, \pi^+ = 4]$. This leads to slightly worse accuracy, $R^* = 0.87, R^\Omega = 0.91$, than in Exp. 1. In Exp. 3, we then tried $[\pi^- = 4, \pi^+ = 5]$ and obtained accuracy of $R^* = 0.87, R^\Omega = 0.90$. Although we do not yet have the exact values reported, we observed anecdotally that the fraction of pixels represented at the higher levels was larger for the two conditions which had the wider π bounds. Thus, even though the accuracies are lower in Exps. 2 and 3, more information is being stored.

We then tried $[\pi^- = 5, \pi^+ = 5]$ and found a significant increase in accuracy to

$R^* = 0.96, R^\Omega = 0.97$. However, we also saw that in this condition, on average, far fewer L1 macs became active on each frame: thus, far fewer pixels were represented at the higher levels (and thus far less information stored). Finally, we tried $[\pi^- = 3, \pi^+ = 3]$, which resulted in accuracy of $R^* = 0.94, R^\Omega = 0.96$. The numbers of active L1 macs per frame was much greater than in Exp. 4 and comparable to Exp. 1. This study supports our belief that having tighter π bounds per se is better than having looser bounds, but to be more certain, in future studies, we will add more systematic tabulation of the fraction of pixels represented at the higher levels, which will allow us to judge the relative information storage capacity between models and thus make a better overall judgment.

Table 12. Recognition Accuracy Results for Section 5.9 Studies

Exp	Lev	M	Q	K	Ave. U-RF	Ave π	Num. Snips.	Num. Weights	Class. Acc.	Ave			
										R^*	R^Ω	R^*	R^Ω
1	6	1	6	6	9	[1,9]	15	7401474	14/15	1.0	1.0	0.93	0.95
	5	9	6	6	5	[2,3]				0.99	0.99		
	4	25	7	7	6	[2,3]				0.95	0.97		
	3	64	7	7	5	[1,2]				0.91	0.93		
	2	121	7	7	5	[1,1]				0.89	0.93		
	1	256	7	7	14	[4,4]				0.84	0.87		
2	6	1	6	6	9	[1,9]	15	7401474	14/15	1.0	1.0	0.87	0.91
	5	9	6	6	5	[2,3]				0.94	0.96		
	4	25	7	7	6	[2,3]				0.92	0.96		
	3	64	7	7	5	[1,2]				0.85	0.92		
	2	121	7	7	5	[1,1]				0.82	0.88		
	1	256	7	7	14	[3,4]				0.69	0.74		
3	6	1	6	6	9	[1,9]	15	7401474	15/15	0.99	0.99	0.87	0.90
	5	9	6	6	5	[2,3]				0.98	0.98		
	4	25	7	7	6	[2,3]				0.91	0.95		
	3	64	7	7	5	[1,2]				0.83	0.89		
	2	121	9	9	5	[1,1]				0.81	0.88		
	1	256	9	9	14	[4,5]				0.69	0.70		
4	6	1	6	6	9	[1,9]	15	7401474	15/15	1.0	1.0	0.96	0.97
	5	9	6	6	5	[2,3]				1.0	1.0		
	4	25	7	7	6	[2,3]				0.97	0.99		
	3	64	7	7	5	[1,2]				0.95	0.97		
	2	121	11	11	5	[1,1]				0.95	0.96		
	1	256	9	9	14	[5,5]				0.91	0.91		
5	6	1	6	6	9	[1,9]	15	7401474	15/15	1.0	1.0	0.94	0.96
	5	9	6	6	5	[2,3]				0.99	0.99		
	4	25	7	7	6	[2,3]				0.94	0.97		
	3	64	7	7	5	[1,2]				0.92	0.95		
	2	121	11	11	5	[1,1]				0.92	0.96		
	1	256	9	9	14	[3,3]				0.88	0.91		

Also, we must emphasize that in this study, all we did was vary the π bounds of the L1 macs while keeping all other parameters, in particular, the π bounds of the macs at other levels, and the U-RF sizes at all levels constant. We strongly believe that maximizing storage capacity at the same time as maximizing recognition accuracy will entail simultaneous movement through many of these parameters in coordinated. We will focus on such larger-scale parameter searching in the next reporting periods, under Tasks C, D, E, and H.

5.9.1. Effect of U-RF Sizes/Overlaps, and π -bounds on Capacity/Accuracy

The model in this study had 712 macs across 6 internal levels, with 400 L1 macs, 196 L2 macs, etc. (see Figure 43), and with a total of from ~16.1 to 17 million weights. The [train-test set](#) contained $N=45$ 8-frame 32x32 snippets. The baseline experiment, E1, had small Q and K values at all levels and the same saturation threshold, 0.30, for all afferent matrices at all levels. The detailed saturation levels attained are reported as well as the recognition accuracy which was about 60% averaged across all levels.

Table 13. Recognition Accuracy Results for this Study

								$\Omega_U, \omega_U, \% \text{ U-RFs sat.}$						Ave
E	L	M	Q	K	$\langle Z_U \rangle$	Ave π	W (x10 ⁶)	U	H	D	Λ	C	R^*, R^Ω	R^*, R^Ω
1	6	1	6	6	9	[1,9]	16.14	.3,0.31,100	.3,0.3,100	NA	70.3	12	.39, .39	0.60, 0.61
	5	9	6	6	8	[2,3]		.3,0.3,100	.3,0.31,100	.3,0.31,100	69.3		.74, .72	
	4	25	7	7	12	[4,4]		.3,0.29,80	.3,0.31,100	.3,0.29,60	71.5		.77, .77	
	3	81	7	7	9	[3,3]		.3,0.26,33	.3,0.3,100	.3,0.23,11	76.8		.68, .71	
	2	196	7	7	13	[3,4]		.3,0.27,60	.3,0.3,100	.3,0.23,50	85.2		.58, .61	
	1	400	7	7	17	[4,5]		.3,0.31,100	.3,0.26,50	.3,0.25,35	86.0		.46, .44	
2	6	1	6	6	9	[1,9]	16.14	.3,0.3, 100	.3,0.3, 100	NA	70.3	13	.44, .44	0.59, 0.60
	5	9	6	6	8	[2,3]		.3,0.3, 100	.3,0.31, 100	.3,0.31, 100	69.3		.73, .72	
	4	25	7	7	12	[4,4]		.3,0.3,55	.3,0.3, 100	.3,0.26,50	71.5		.75, .75	
	3	81	7	7	9	[3,3]		.3,0.26,50	.3,0.3, 100	.3,0.20,15	76.8		.66, .70	
	2	196	7	7	13	[3,4]		.3,0.29,90	.3,0.3, 100	.3,0.25,25	85.2		.55, .58	
	1	400	7	7	17	[4,5]		.25,0.26,100	.3,0.26,50	.3,0.25,85	86.0		.42, .40	
3	6	1	6	12	9	[1,9]	16.97	.3, 0.24,0	.3, 0.26,0	NA	70.3	38	.85, .85	0.68, 0.70
	5	9	6	10	8	[2,3]		.3, 0.16,0	.3, 0.24,0	.3, 0.23,0	69.3		.84, .84	
	4	25	7	10	12	[4,4]		.3, 0.22,0	.3, 0.23,0	.3, 0.15,0	71.5		.77, .80	
	3	81	7	7	9	[3,3]		.3,0.26,33	.3,0.3,100	.3, 0.17,0	76.8		.66, .71	
	2	196	7	7	13	[3,4]		.3,0.27,60	.3,0.3,100	.3,0.23,21	85.2		.55, .58	
	1	400	7	7	17	[4,5]		.25,0.26,100	.3,0.27,80	.3,0.25,35	86.0		.42, .40	

We saw that R values were poor even at the first internal level, L1, i.e., $R_1^* = 0.46, R_1^\Omega = 0.44$.

Based on manual analysis of results of individual snippets, we saw that crosstalk interference was very high at L1, which not only reduces R values for L1, but for higher levels as well since they depend on the accuracy achieved at L1. We therefore tried reducing the L1 U saturation threshold (Ω_U) from 0.30 to 0.25 in E2 (highlighted in red). We reasoned that this would shut down learning in the U wts to L1 at an earlier time during learning and thus lower the average U summations to L1 cells during recognition, thus reducing crosstalk. However, as seen in Table 13, this actually worsened performance at L1 and L2, i.e., $R_1^* = 0.42, R_1^\Omega = 0.40$,

$R_2^* = 0.55, R_2^\Omega = 0.58$. R values at higher levels remained about the same. We then realized that

simply shutting down U learning from L0 to L1 at an earlier point in the training run simply prevents any L0-to-L1 U associations from being made following the frame on which an L1 mac's U-RF freezes. Clearly, during recognition tests, an L1 mac cannot reliably reinstate L1 codes that were activated on frames following freezing during learning.

We then noticed that all of the RFs (U, H, and D) at higher levels, in particular at L6, on which the classification decision depends, were frozen (violet values in the table), and in fact, froze relatively early during learning. In particular, the L6 mac's U-RF froze during presentation of the 17th snippet, meaning that no L5-L6 U associations were learned from the 18th snippet on. This greatly reduced R values and made correct classification of snippets 18 and higher, impossible. Therefore, to reduce the rate of saturation of higher-level matrices, the model of E3 has increased K values in the three highest levels. As can be seen in the table, this had the desired effect of preventing higher level matrices from saturating and thus increasing R values at those levels, including a huge increase at L6 from $R_6^* = 0.44, R_6^\Omega = 0.44$ to $R_6^* = 0.85, R_6^\Omega = 0.85$. Also, 38 of 45 snippets were correctly classified. These K increases did increase the number of weights, W , from ~16.1 to ~17 million. However, generally speaking, increasing K or Q values at higher levels, where the numbers of macs (and thus, cells) are small is much more tolerable than increasing K or Q values at lower levels, e.g., L1 and L2, which greatly increases W .

We underscore that the property discussed in the introduction that recognition accuracy generally increases with level is clearly seen in these results. We provide detailed results in Section 6.7 that show the two principles giving rise to the property.

The model used in the study of Table 13 reflects some parametric architectural improvements from the model reported on in TR17. Over the course of manual searching of parameter space following the TR17 experiments, we determined that having larger degrees of fan-in from one level to the next, or in other words, slightly larger U-RF sizes, allowing greater overlap of U-RFs, while at the same time tightening the π bounds for all levels, seemed to result in both: a) greater representation of input features at all higher levels; and b) greater recognition accuracy at higher levels. Figure 44 shows the gross architectural differences between the 476-mac TR17 model and the newer 712-mac model. The model of experiments E4-E6 (Figure 44a) had a total of 476 macs and smaller and less overlapped U-RFs than those of the model used in E1-E3 and E7, which had a total of 712 macs (Figure 44b). Small samples (~5-10%) of the weights comprising the U-RFs of several cells at different levels are shown with blue sprays: these sprays denote the “fan-in” of U connections.

In addition, we determined that we can follow a level-by-level parameter search/optimization process starting from L1 and proceeding upward. This has led, thus far, to the 712-mac model reported on in Table 13. We say “thus far” because although the combination of numbers of macs per level, U-RF sizes, and π bounds in this model yield markedly better results than those used in the model of TR17, which had only 476 macs, we do not know how close we are to the optimal parameters.

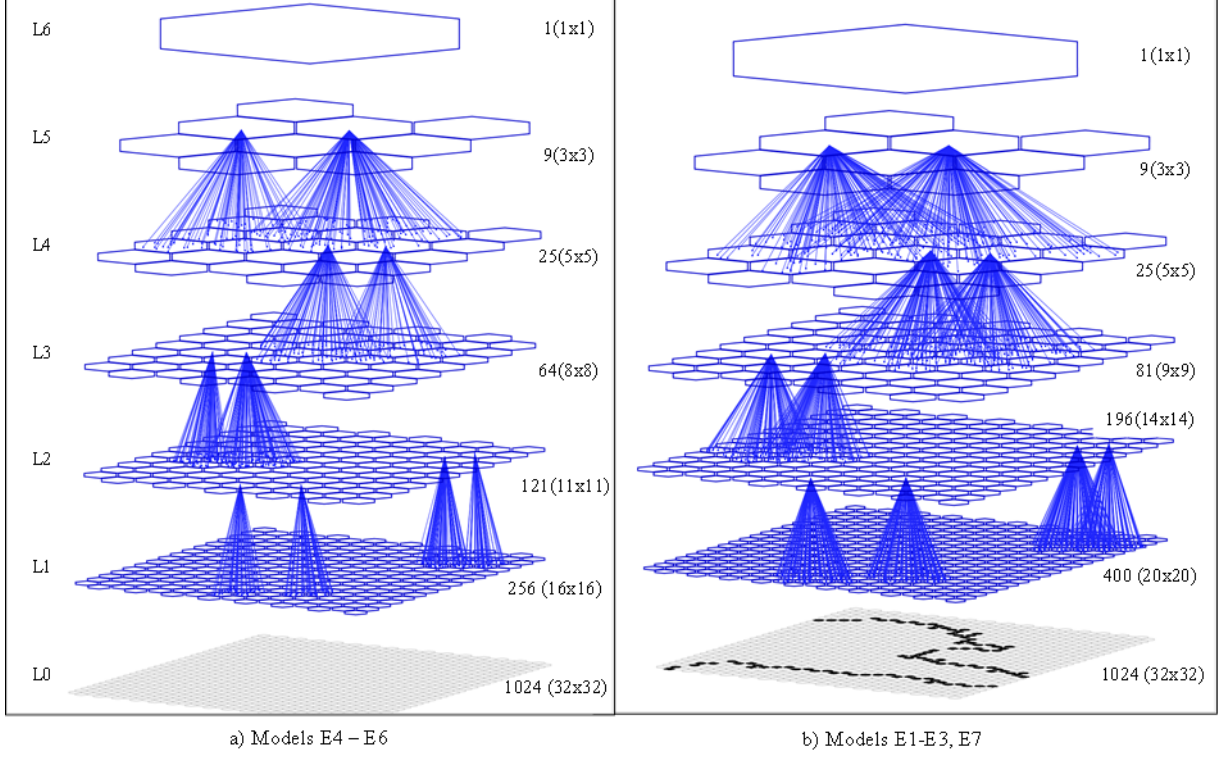


Figure 44: Gross Architectures of Models Used in Studies of Table 13-Table 15

We performed the following study to show directly the improvement in recognition accuracy and other measures in moving by one particular path of parametric variation from the TR17 model, reported as E4 in Table 14 to the model of E3 in Table 13. We describe motivations for the specific parameter changes and main resulting differences from one experiment to the next below.

Because of its higher Q and K values at L1 and L2, the E4 model has many more weights than the E3 model, ~ 23.5 vs. ~ 17 million. Despite this, the E4 model achieves only $R^* = 0.53$, $R^\Omega = 0.53$. Based on some of our findings in experiments, E1-E3, we made two modifications to E4. First, given that the R values of higher levels and classification accuracy are reasonably good despite the low L1 and L2 R values ($R_1^* = 0.42$, $R_1^\Omega = 0.40$) in E3, we decided that we could lower the L1 and L2 Q and K values slightly, as shown, to greatly lower W , without decreasing attainable R values at higher levels. Second, seeing that higher level matrices were also frozen (as in E1 and E2), we increased Q and K values at higher levels to match those of E3. Therefore, the essential difference between E5 and E3 is in the U-RF sizes, π bounds, and degrees of overlap between neighboring macs' U-RFs (as illustrated in Figure 44) and about 2.8 million fewer weights. As expected, these changes led to reduced R values in the lower levels, e.g., $R_1^* = 0.54$, $R_1^\Omega = 0.54$, but significantly better overall (i.e., R values averaged across all levels) performance, $R^* = 0.58$, $R^\Omega = 0.59$ despite having ~ 9 million fewer weights than the E4 model.

Table 14. Recognition Accuracy Results for this Study

								$\Omega_U, \omega_U, \% \text{ U-RFs sat.}$					Ave
E	L	M	Q	K	$\langle Z_U \rangle$	Ave π	W (x10 ⁶)	U	H	D	C	R^*, R^Ω	R^*, R^Ω
4	6	1	6	6	9	[1,9]	23.47	.3,0.3,100	.3,0.3,100	NA	11	.36, .36	0.53, 0.53
	5	9	6	6	5	[2,3]		.3,0.3,100	.3,0.31,100	.3,0.31,100		.51, .51	
	4	25	7	7	6	[2,3]		.3,0.3,100	.3,0.3,100	.3,0.3,95		.56, .55	
	3	64	7	7	5	[1,2]		.3,0.26,50	.3,0.3,100	.3,0.29,90		.57, .58	
	2	121	11	11	5	[1,1]		.3,0.29,0	.3,0.3,100	.3,0.29,50		.61, .63	
	1	256	9	9	14	[4,4]		.3,0.31,100	.3,0.1,0	.3,0.11,0		.58, .58	
5	6	1	6	12	9	[1,9]	14.23	.3,0.27,0	.3,0.26,0	NA	33	.73, .73	0.58, 0.59
	5	9	6	10	5	[2,3]		.3,0.23,0	.3,0.29,50	.3,0.28,20		.59, .61	
	4	25	7	10	6	[2,3]		.3,0.29,55	.3,0.24,40	.3,0.23,0		.55, .58	
	3	64	7	7	5	[1,2]		.3,0.27,50	.3,0.3,100	.3,0.3,50		.51, .53	
	2	121	9	9	5	[1,1]		.3,0.21,0	.3,0.3,100	.3,0.28,50		.53, .56	
	1	256	8	8	14	[4,4]		.3,0.31,100	.3,0.11,0	.3,0.14,0		.54, .54	
6	6	1	6	12	9	[1,9]	17.10	.3,0.27,0	.3,0.27,0	NA	38	.84, .84	0.64, 0.67
	5	9	6	10	5	[2,3]		.3,0.24,0	.3,0.3,80	.3,0.29,20		.70, .72	
	4	25	7	10	6	[2,3]		.3,0.25,40	.3,0.24,33	.3,0.22,0		.63, .67	
	3	64	9	9	5	[1,2]		.3,0.25,50	.3,0.3,100	.3,0.15,5		.56, .61	
	2	121	9	9	5	[1,1]		.3,0.24,0	.3,0.3,100	.3,0.28,50		.57, .61	
	1	256	8	8	14	[4,4]		.3,0.31,100	.3,0.11,0	.3,0.14,0		.55, .56	

We then changed the E5 model in two ways to get to the E6 model. We simply increased the Q and K values for the L3 macs. This added almost 3 million weights, giving the E6 and E3 models comparable total weights (W). It increased accuracy to $R^* = 0.64, R^\Omega = 0.67$, substantially lower than that of E3, $R^* = 0.68, R^\Omega = 0.70$. Given the comparable W values, the equal numbers of levels, and the equal Q and K values for levels L4-L6, we conclude that the differences in the U-RFs sizes, $\langle Z_U \rangle$, U-RF overlaps, and π bounds between E3 and E6 (as seen in Figure 44) are important. Specifically, we observe that by:

1. making the U-RF sizes larger,
2. making the average overlaps between neighboring U-RFs larger,
3. at the same time, maintaining or reducing the fraction of afferent U features that needs to be active in order for macs to activate,
4. and tightening the bounds on the number of active features needed to activate a mac,

the model represents more of the information in the input set and achieves significantly better recognition accuracy. The above list of modifications begins to look like a general “law” for designing hierarchical compositional networks. However, much parametric testing remains. For example, throughout the recent studies of TR17 and this report, we have set the H-RFs at all levels to include only the single target mac, i.e., H wts only connect cells within a single mac.

Perhaps making the H-RFs larger and overlapped at one or more of the levels would further increase performance. Similar considerations apply to the D wts as well. These, and many other manipulations need to be explored before we can feel confident that we understand the optimal policies for wiring up a hierarchical information processing network.

In the last experiment of this section, E7 in Table 15, we took the E3 model and substantially increased the number of D-wts, so that the D projections approximately mirrored the U projections. That is, we set parameters so that in most cases, if an L_J mac projected to an L_{J+1} mac, then that L_{J+1} mac also projected to that L_J mac: we did not expressly control for this in prior studies and for the most part the sizes of the D-RFs were smaller than of the reciprocal U-RFs. Correcting for this added about 3.4 million weights relative to E3. While this significantly increased R values at L3 and L4 (comparing E7 to E3, blue-highlighted text in Table 15), R values at higher levels were unchanged and the average R values over all levels were only slightly improved. Again, we have a great deal more parametric exploration to do in order to understand, in general, how best to set all of these myriad interacting parameters to maximize performance / capacities. In particular, we do not draw any strong conclusion from this null result of Experiment E7.

Table 15. Recognition Accuracy Results for this Study

E	L	M	Q	K	$\langle Z_U \rangle$	Ave π	W (x10 ⁶)	$\Omega_U, \theta_U, \% \text{ U-RFs sat.}$			Λ	C	R^*, R^Ω	Ave R^*, R^Ω
								U	H	D				
7	6	1	6	12	9	[1,9]	20.4	.3,0.23,0	.3,0.26,0	NA	70.3	35	.84, .84	0.70, 0.72
	5	9	6	10	8	[2,3]		.3,0.16,0	.3,0.23,0	.3,0.23,0	69.3		.84, .84	
	4	25	7	10	12	[4,4]		.3,0.22,0	.3,0.23,0	.3,0.15,0	71.5		.80, .83	
	3	81	7	7	9	[3,3]		.3,0.26,33	.3,0.3,100	.3,0.17,0	76.8		.70, .77	
	2	196	7	7	13	[3,4]		.3,0.28,62	.3,0.31,10	.3,0.23,20	85.2		.58, .63	
	1	400	7	7	18	[4,5]		.25,0.26,100	.3,0.26,50	.3,0.25,35	86.0		.42, .41	

Table 16. Symbol Definitions for this Section

Symbol	Definition	Symbol	Definition
E	Experiment number.	L	Hierarchical level of the model
M	Number of macs in Level	Q	Number of CMs per mac
K	Number of cells per CM	N	Num. of snippets in train / test set
$\langle Z_U \rangle$	Ave. number of afferent U features (pixels for L1 macs, subjacent macs for L2 macs and higher) to a mac, i.e., aver. U-RF size.		
Ave. π	π range consisting of the ave. π^- and ave. π^+ over all macs of a given level, i.e., $\left[\langle \pi^- \rangle, \langle \pi^+ \rangle \right]$		
$Z_U(\cap^1)$	Ave. U-RF overlap of adjacent macs. “ \cap^1 ” indicates RF intersection with immediately adjacent, i.e., distance “1”, neighbors		
Λ	Fraction of input features (pixels) represented at a given level.		
W	Total num. of wts in network	ω_U	Actual ave. fraction of increased weights in U-RFs of macs of a given level
Ω_U	U-RF saturation threshold		% of saturated U-RFs
		C	Classification accuracy, i.e., number of correctly classified snippets
R^*	Ave. trace accuracy over all snippet frames	R^Ω	Trace accuracy on last snippet frame

6. DISCUSSION

The major accomplishments achieved during this research project, which in concert underlie our successful demonstrations of feasibility, are summarized in this list and throughout the section.

1. We have created a general model allowing
 - a. an arbitrary number of hierarchical levels
 - b. communication within between adjacent levels (and between non-adjacent levels)
 - c. each level can consist of many macs
 - d. each mac stores information using sparse distributed representations (SDRs)
2. Demonstrated that the model can learn and retrieve a large number of complex spatiotemporal binary input patterns. Learning is single-trial and learning time for each new frame remains constant through life of system, i.e., regardless of how many frames are stored. Retrieval time of best-matching frame also remains constant for life of system.
3. Demonstrated near SOA classification performance on Weizmann video event recognition data set, while demonstrating world-leading learning time.

6.1 Regarding Sparsey's Information Storage Capacity

In earlier, related studies of Sparsey, model instances achieved approximately 0.1 bits/synapse, which is well within an order of magnitude of the theoretical limit for associative memories, of ~ 0.69 bits/synapse (Willshaw, Buneman et al. 1969). Two remarks are relevant here.

First, even if the model's storage capacity (essentially equivalent to space complexity) asymptoted at 0.1 bits/synapse, this would be sufficient to scale to the kinds of massive "big data" problems of interest to DARPA and business world in general. For example, a Sparsey instance with 100 billion synapses would be able to store 10 billion bits of information.

Crucially, all of this information would be *instantly* retrievable in closest-match mode, or in other words, in context-dependent fashion.

The potential impact can perhaps be more readily appreciated in the terminology of conventional databases. Sparsey will be able to automatically store, on-the-fly (i.e., in a streaming mode), a large number (N) of records, each having a large number (M) of fields, and subsequently permit arbitrary, *unanticipated many-field* queries, all of which are returned in the same constant time. In the world of conventional databases, achieving even $\log N$ (let alone, constant) time on any particular m -field query requires the prior construction of an index that is specific to the particular m fields in that query. Thus, guaranteeing $\log N$ response time to all queries up to m^{th} -order would require prior construction of *all* ${}_M C_m$ indexes. What we are saying is that, due to its SISC (similar-inputs-to-similar-codes) property, Sparsey automatically discovers the subset of these ${}_M C_m$ queries that would reveal some interesting structure of the input space and *effectively*

creates an index for each such query—up to some, probably low-moderate⁸, m value—as a by-product of the simple act of storing (building hierarchical SDRs for) the individual items as they are presented. Unlike the case of conventional databases, *no* additional time or work is needed for the construction of these “indexes”; they are implicit in the pattern of synaptic weights. And, just as importantly, there is no need to predefine at the outset the entire set of queries that we can anticipate may someday be useful. That set is in fact implicit in the statistical structure of the data and is what is automatically discovered and embedded in the synaptic weights during storage. Finally, and as noted above, the retrieval time for any of these queries is *constant*, not $\log N$ as for state-of-the-art tree-based indexes used in existing databases.

Second, we believe it is very likely that we can get close to the theoretical limit (~ 0.69 bits/synapse) by: a) using hierarchical models with many levels; and b) other techniques / mechanisms, e.g., enhancements to Sparsey’s learning model. By point a, we mean that hierarchical data structures can compress information exponentially relative to flat data structures. As an example, consider the game of Twenty Questions, qua data structure: that first question, “Animate vs. Inanimate”, a binary question, discounts half of the universe of possible answers. The information content of that question must be very much larger than one bit. Regarding point b, one potential improvement would be to make synaptic increases made to/from cells that have just been activated be larger than increases to/from synaptic cells that have been active for a number of time steps (more generally, a gradient of increase levels as a function of number of time steps active). This could be exploited to improve the recognition accuracy on early frames of a snippet, and thus the overall accuracy over a whole test set. Many variations on this theme are possible.

6.2 Supervised Learning via Cross-Modal Unsupervised Learning

In general, the categories in the physical world are smooth (linear) in the neighborhood around any single sample but possibly very nonlinear and intertwined with other classes at the global scale [cf. (Saul and Roweis 2002)]. This is illustrated with respect to the abstract, high-dimensional visual space on the left of Figure 45. Lowercase “a” is visually more similar to lowercase “e” than to uppercase “A”. Yet “a” and “A” are in the same category, while “e” is not. At right of Figure 45, we show a notional V1 mac to which we will assume the abstract visual space is mapped. Sparsey handles the local smoothness of natural categories because it has the SISCs property. The formation of the mapping from “a” to ${}^{V1}\phi_a$ induces a hyperspherical region in the high-dimensional input space within which other inputs will be mapped to ${}^{V1}\phi_a$ with a certain probability (which depends on multiple model parameters).

However, by the same SISC property, the code to which “A” is mapped, ${}^{V1}\phi_A$, will (statistically) have lower intersection with ${}^{V1}\phi_a$ than the code to which “e” is mapped, ${}^{V1}\phi_e$, as shown in Figure 45 (light blue cells are the intersection of ${}^{V1}\phi_e$ and ${}^{V1}\phi_a$; green cells are intersection of ${}^{V1}\phi_A$ and ${}^{V1}\phi_a$). So, how can Sparsey physically represent the fact that “a” and “A” are in the same

⁸ The point is made throughout most leading-edge machine-learning papers, e.g., “Deep Learning”, that the *intrinsic* dimensionality of many naturalistic populations (including those derived from human activity, e.g., the set of all bank records, or all economic databases, etc.) is numbered in the tens or maybe a few hundred, not thousands.

category, while “e” is in a different category? More generally, how can Sparsey address the highly nonlinear and intertwined nature of natural categories at the more global scale?

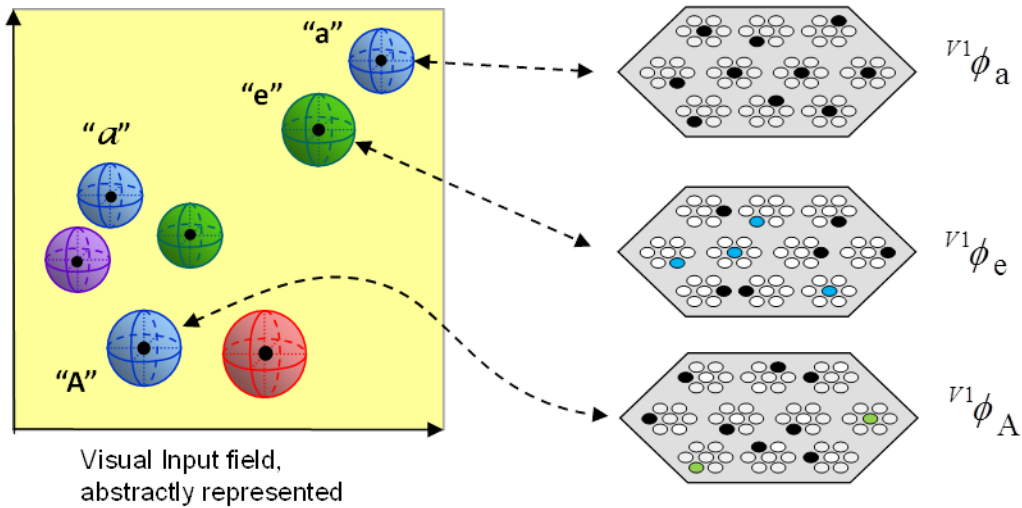


Figure 45: Mapping from High-Dimensional Visual Input Space to Notional V1 Mac

Sparsey’s answer is shown in Figure 46. The solution is to associate codes in one field, e.g., a visual input field, with codes in another field, which represents the *categories* per se, i.e., their names (labels). This essentially implements supervised learning via cross-modal unsupervised, associative learning. Here, the vision (V) modality is a 2-level network, whose top level V1 (a.k.a. L1), corresponds to the mac of Figure 45. The abstract input space of Figure 45 is now depicted concretely as the hexagonal “aperture”, V0” (a.k.a. L0), which consists of ~140 binary pixels. The “Symbol (S) Modality” is a 1-level network, i.e., simply an input level, “S0”, which uses a localist representation of symbols (class labels), and which is fully connected to V1.

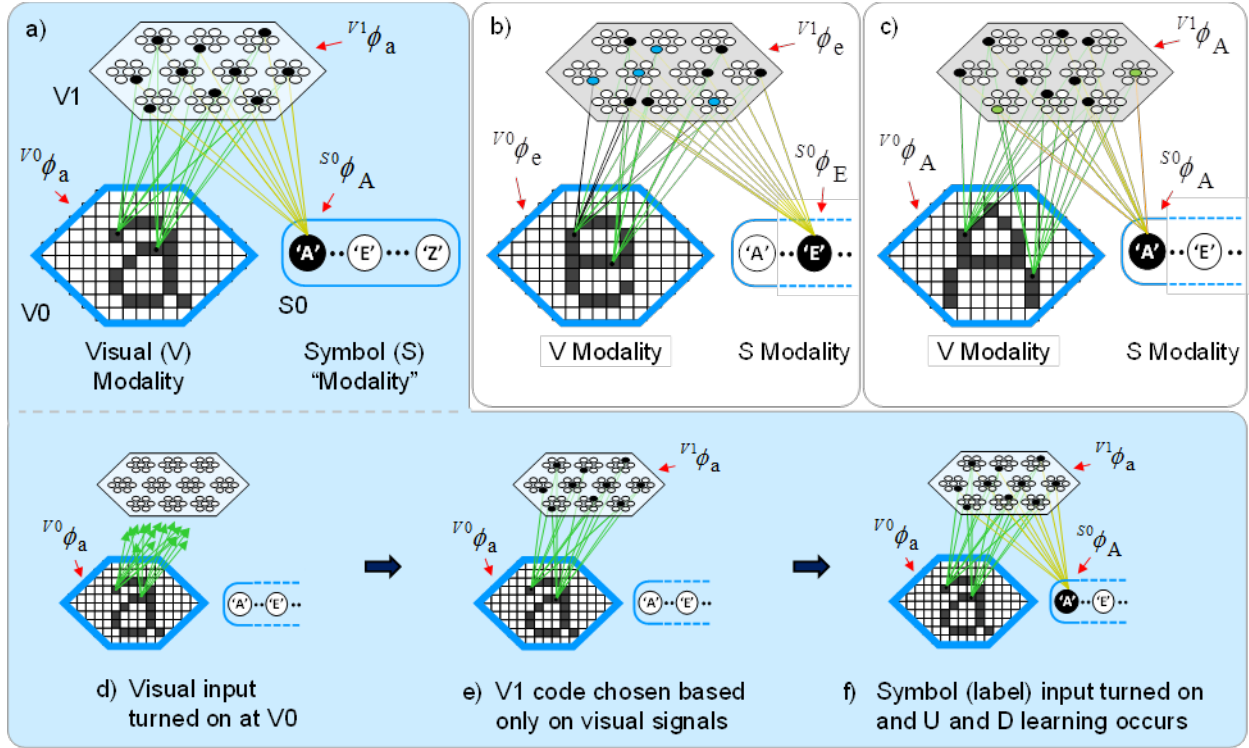


Figure 46: Implementation of Supervised Learning as an Instance of Cross-Modal Unsupervised Association

In Figure 46a, we present input “a”: we name that input pattern, $V^0\phi_a$, where the lead superscript means that it is a pattern in V0 (biologically, V0 corresponds to visual thalamus). The U signals from V0 to V1 cause an SDR code, $V^1\phi_a$, to be activated in V1. Once activated, U and D learning occurs between $V^0\phi_a$ and $V^1\phi_a$. The green lines mean that these weights are being increased for the first time, i.e., from 0 to 1. Contemporaneously, we activate, as a supervisory signal, the representation of class ‘A’, $S^0\phi_A$, in the symbol input field, S0. However, U signals from S0 to V1 *do not influence* the choice of winners in V1. Rather, activation of a symbol unit on a learning trial simply serves to indicate which weights to increase. In Figure 46a, these would be the U and D weights between $S^0\phi_A$ and $V^1\phi_a$, which are depicted with a different color (yellow) to indicate the different semantics of these *inter-modal* weights. Thus, the internal SDR code, $V^1\phi_a$, becomes associated, in a single-trial, with the class ‘A’. Figure 46d-f make the causal ordering of steps more explicit, emphasizing that the V1 code is chosen based only on the visual signals.

Figure 46b shows a similar scenario in which presentation of “e” gives rise to code, $V^1\phi_e$, which is then associated with $S^0\phi_E$. The black U weights indicate weights increased on a prior association; here, in Figure 46a. In Figure 46c, presentation of “A” gives rise to a new V1 code, $V^1\phi_e$, which (consistent with the SISC property) has less overlap with $V^1\phi_a$ (the two light green units) than $V^1\phi_e$ did. Nevertheless, we activate the symbol representation, $S^0\phi_A$, thus associating, again, with a single trial, this visual pattern (“A”), which looks completely different from “a”, with the same class as “a”.

This general cross-modal association framework allows arbitrarily different inputs in one modality, e.g., visual, to be associated with the same category, thus building up via single-trial learning, categories with arbitrarily nonlinear boundaries, in particular, boundaries that are formally unions of hyperspheres. In fact, there is substantial evidence that children learn new words, and thus their corresponding potentially highly nonlinear category boundaries, in this essentially single-trial associative fashion (Carey and Bartlett 1978, Dollaghan 1985, Jusczyk 1999, Behrend, Scofield et al. 2001, Smith and Yu 2008).

We emphasize that this principle—corresponding to what in cognitive psychology is called “exemplar-based categorization” (EBC)—is orthogonal to the principle of hierarchical (and compositional) representation *per se*. They are separate resources providing independent sources of computational efficiency. Sparsey utilizes both principles. The success of the Deep Learning (DL) framework is due to breakthroughs allowing efficient exploitation of hierarchy (Bengio, Courville et al. 2012). But EBC has thus far been largely absent from DL, as well as from mainstream pattern recognition and machine learning approaches. The following quote supports the importance of EBC (given the close relationship of EBC and single-trial learning).

“I think all our favorite [inference] methods – Gibbs sampling, overrelaxation, hybrid Monte Carlo, variational methods, EM, gradient descent – are all too creepy-crawly slow... The world isn’t an adversary. It should be possible to solve many learning problems in a couple of iterations through a reasonable data set, rather than thousands. It may be too much to ask for a one-shot learning method, but maybe we should be looking for one-and-a-half-shot learning algorithms.” – David MacKay in (Frey 1998).

Stepping back, we point out that AI and MI models fall into two classes representing fundamentally different views on categorization, and consequently, on memory and cognition (thinking). In the first class, representations of category boundaries, e.g., hypersurfaces in a representational space, are computed from *sets* of samples and maintained in memory *separate* from the representations of the individual samples themselves. Future input samples are then compared directly to the category representations, not to the samples. Indeed, the representations of the individual samples are no longer needed once the category representations have been built. These explicit category representations have been referred to as abstract categories and this view of categorization, as *abstraction-based*. DL, convolutional nets, and HMAX-family models are in this class.

The second class is the aforementioned EBC, of which Sparsey is an instance. Memory traces, i.e., SDR codes, of individual inputs are permanently retained and future instances of categorization (recognition) entail comparing the input to *all* stored exemplars. However, with SDR, all the exemplars are physically overlapped in memory. Thus, while Sparsey *effectively* iterates over all stored exemplars during the comparison process (during both recognition and learning), it *actually* iterates over representational units (i.e., cells), and more finely, over weights, the numbers of which remain fixed as new traces are stored. Indeed, this underlies Sparsey’s most important and distinguishing property, fixed-time learning and best-match retrieval (recognition).

Figure 47 shows the difference between abstraction-based categorization (ABC) and EBC schemes. As in Figure 45, different colors signify different categories. In the ABC scheme of Figure 47a, samples of a given category are used to compute a single, *connected* boundary – e.g., a hyperplane (dashed lines), a hypervolume (colored ellipses), etc., and the boundaries are used to categorize future samples. However, as Figure 47b shows, the more intermingled the exemplars are, the more nonlinear the category boundaries must be, and the harder it is for gradient-based or sampling-based methods to converge to those boundaries (Bengio and leCun 2007). In contrast, in the EBC scheme of Figure 47c, category boundaries are *implicitly* defined as *unions* of hyperspheres (superimposed on the global category regions). As Figure 47d suggests, with EBC, the difficulty of learning categories remains constant regardless of the degree of intermingling, essentially because category boundaries no longer need to be connected.

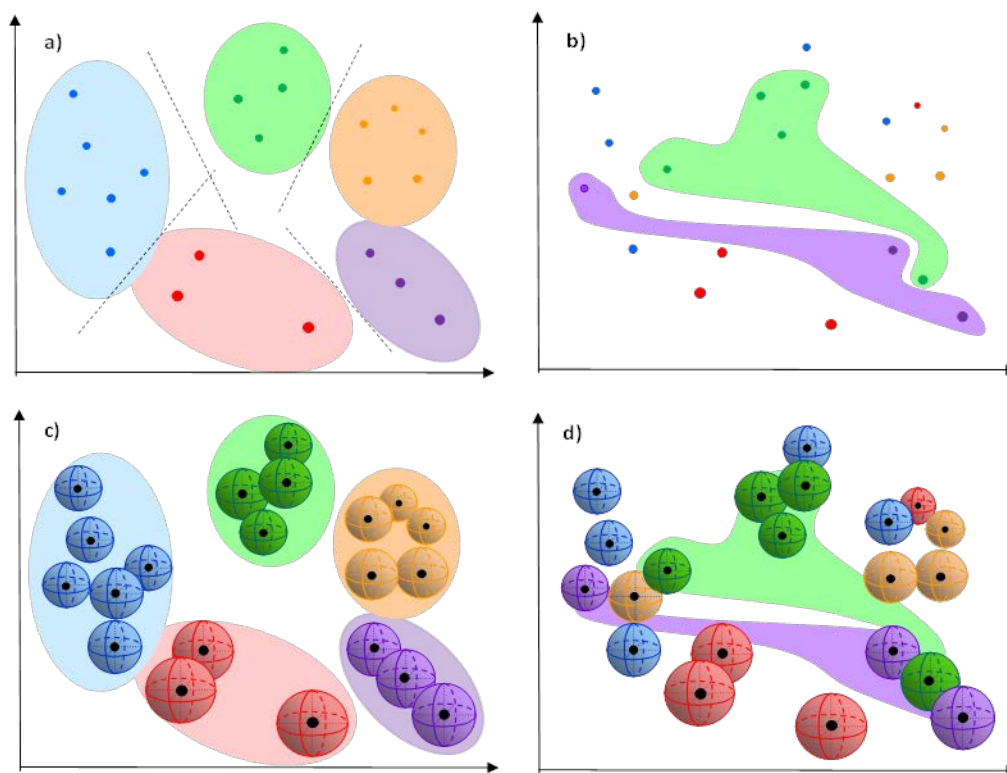


Figure 47: Abstraction-Based vs. Exemplar-Based Categorization Schemes

With regard to an overall pattern recognition system, the act of recognition is only half the story. The other half is the learning process. For ABC systems, this includes the presentation and storage (at least transiently) of all learning exemplars *and construction of the category boundary*. The time complexity of the category boundary construction process is typically the limiting factor. For example, gradient following supervised learning algorithms like Backpropagation and contrastive divergence require numerous epochs (iterations over training set). Similarly, each iteration of the Boltzmann Machine formally requires sampling from the posterior over the representation's state space, which, even when only done approximately, e.g., using Gibbs sampling, is still very slow. However, for EBC systems, learning entails only the presentation and storage of the exemplars. There is no explicit creation of category boundaries. As discussed above, the category boundaries remain implicit in the set of exemplar representations, and in the

case of SDR, the creation time of those representations—i.e., the assignment of mac codes—remains constant for the life of the system.

6.3 Hierarchical Exemplar-Based Categorization

In Technical Report 8, we explained that any single mac in Sparsey implements an exemplar-based (episodic memory-based) form of categorization (cf. Figure 1 of Tech. Report 8). The CSA (which runs in each mac) maps similar inputs to similar (more highly intersecting) mac codes (SISC property). The precise shape of the mapping depends on many parameters (described elsewhere). In particular, parameters can be set so that novel exemplars that are within distance, D , of a stored exemplar, reactivate the exact code of the stored exemplar, with probability, P . The hyperspheres in Figure 48 correspond to this parameter-dependent diameter, D . In a more veridical representation, these hyperspheres would be depicted as hyperspherically (or perhaps hyper-ellipsoidally) symmetric probability density clouds with distance dependent fall-off from the center.

Each mac assigns SDR codes to individual exemplars (samples) that occur (with sufficient regularity) in its input space (until its afferent weight matrices reach a saturation threshold which causes the mac to “freeze”, i.e., to prevent any further codes from being assigned). However, the spatial and temporal scale of the features increases with level, so that as a whole, the network implements a hierarchically nested clustering. This figure shows the abstract representation of the 36-Dimensional input space (6x6 binary pixels) of one particular L1 mac, $M_{(2,0)}^1$, where superscript denotes level, and subscript denotes (row,col) of the mac within the level. Hyperspheres represent several specific inputs (exemplars) that have been stored in $M_{(2,0)}^1$.

In the middle of Figure 48 is a detailed view showing the specific code active in one particular L1 mac, $M_{(2,0)}^1$. The dashed arrow shows the correspondence between that particular code and input instance 1 of the 45° edge class in the abstract space. Small numbers beside input patterns in the abstract space simply serve to identify specific patterns. At lower left, we show that different instances of a class, e.g., 45° edges, can have very low pixel-wise overlap, whereas instances of different classes can have larger overlaps. Thus, instance 1 of the 45° edge class is depicted as closer in hyperspace to the noisy instance of the 0° class than to instances 2 and 3 of the 45° edge class. Due to the SISC property, the codes for these 45° edge instances will have smaller intersection with the code of 45° edge instance 1 than with codes of instances of other classes, e.g., the 0° and 90° instances in this example.

visual space. Remember that in our model, these pixels are binary and the preprocessing turns input color/grayscale video frames into binary edge-filtered frames where all edges are eroded to be 1-pixel wide. At lower left, we illustrate similar principles as in Figure 48 regarding the relation of overlap in input space and functional class membership, but at L2's higher spatial scale.

While we can show the inputs to $M_{(1,0)}^2$ as patterns over an entire 12x12 pixel array, it is important to realize that that input space is formally composed of four separate contiguous (non-overlapping) smaller spaces, i.e., the 6x6 pixel arrays corresponding to $M_{(1,0)}^2$'s four afferent L1 macs. Suppose for simplicity that all four of these L1 macs have been frozen. That is, their afferent matrices from L0 have crossed a saturation threshold such that no further codes will be stored in those macs, i.e., they will undergo no further learning.⁹ Suppose for simplicity that 100 codes (i.e., basis vectors) have been stored in each of those four L1 macs. That means that the (bottom-up) input space for $M_{(1,0)}^2$ is formally 4-D, and specifically, 100^4 . Because we are assuming the L1 macs are frozen, there are now exactly 100^4 unique input patterns that can ever occur to $M_{(1,0)}^2$. This is a vanishingly small fraction of all possible 12x12 binary pixel patterns that could occur. But, even this vanishingly small fraction of all possible 12x12-pixel patterns, $100^4 = 10^8$, is still orders of magnitude greater than an L2 mac, even one of biologically realistic size, i.e., $Q=70$ and $K=20$, could safely store. Perhaps an L2 mac might be able to safely store a few thousand codes at most. But the question is: how big a basis set does a typical L2 mac, which sees only a 12x12-pixel aperture of visual space need in order to adequately represent all future patterns that occur in that aperture? We show just a handful of possible 12x12-pixel patterns that might be contained in $M_{(1,0)}^2$'s basis set. Some of these might appear to be rather random or arbitrary; we will address this point later.

⁹ At least, no further long-term learning. Short-term synaptic weight modification, e.g., representing a working memory concept, might still occur (though that is not currently implemented in the model).

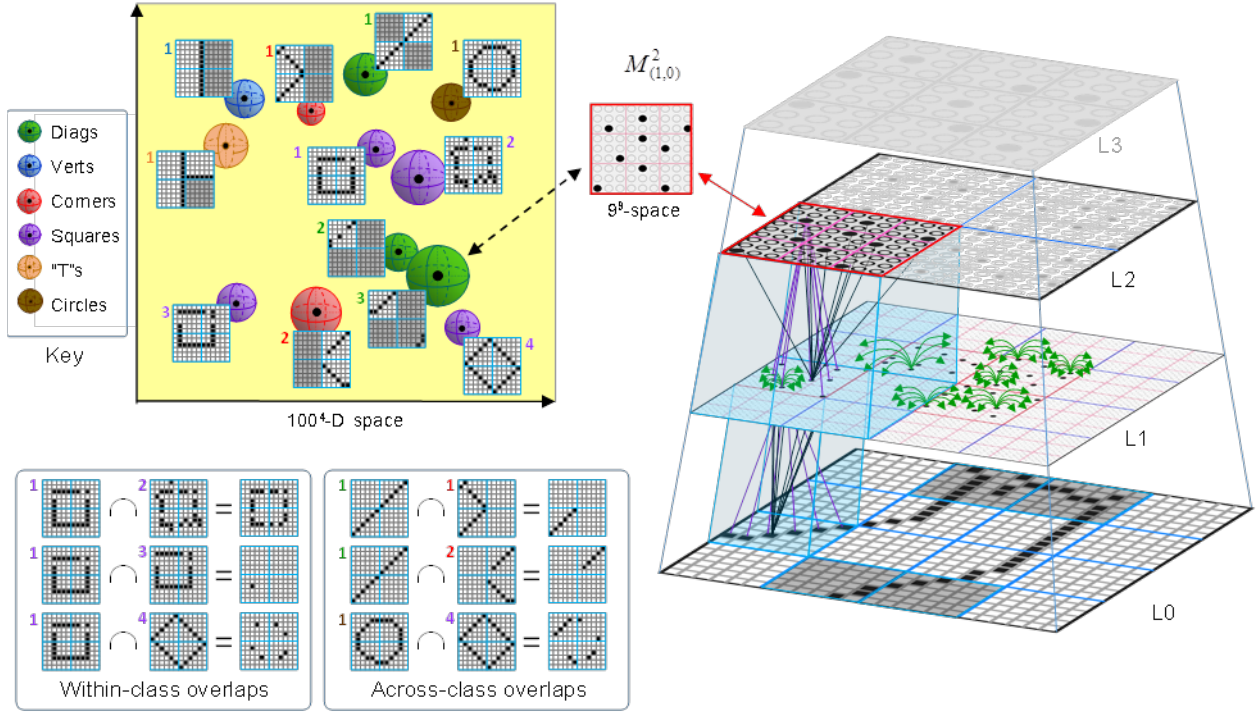


Figure 49: Network of Figure 48 but Emphasizing One Particular Active L2 Mac

Now let's consider how each of these four input dimensions to $M^2_{(1,0)}$ (outlined in red in Figure 49) is ordered. As we have noted, each of those four dimensions has exactly 100 unique values. Each of these values corresponds to a particular 6x6 pattern, a tiny representative sample of which can be seen in the abstract space at upper left of Figure 49. In the abstract input space depiction at left, 6x6 patches are shaded if either too few or too many pixels are active for the corresponding L1 mac to have become active. See text for further discussion. Note that the values themselves are not pixel patterns, but SDR codes, in this case, sets of $Q=9$ binary cells, one per CM. How are these 100 *codes* ordered along a *dimension*? Quite simply, because the mapping from pixel space to L1 code space preserves similarity, the L1 codes are naturally ordered in terms of intersection size, i.e., codes with higher intersections represent more similar 6x6-pixel patterns.¹⁰ And, since the mappings from each L1 mac to $M^2_{(1,0)}$ also preserve similarity, the overall mapping from input (L0) space to L2 code space also preserves similarity. However, the similarity metric is compositional. That is, $M^2_{(1,0)}$'s measure of the similarity, 2G (introduced in prior technical reports), of a novel input decomposes as the product of the similarities, 1G measured in each of its afferent L1 macs.

¹⁰ For simplicity, we are ignoring the fact that in actuality, Sparsey's mac codes are temporal context dependent. That is, each code stored in a mac actually represents not just the purely spatial (bottom-up, U) input to the mac, but the specific *moment*, i.e., the spatial input in the temporal (sequential) context of the frames leading up to the current spatial input.

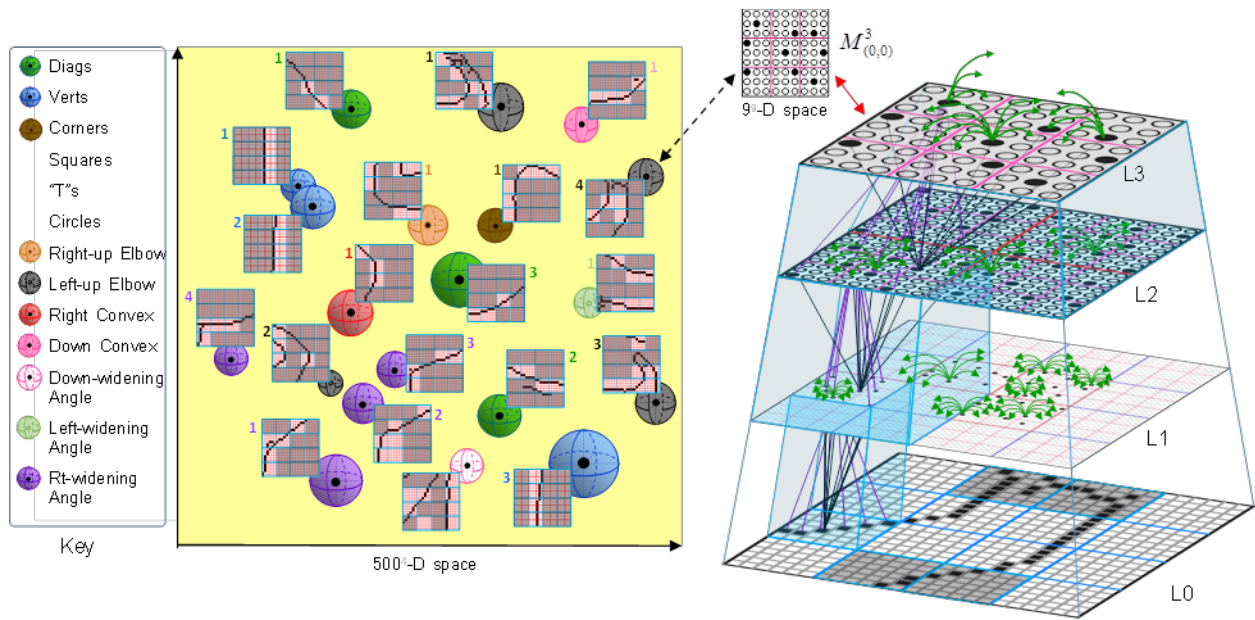


Figure 50: Network of Prior Figures but Emphasizing the L3 Mac

The 24x24 pixel patterns shown in the abstract space at left of Figure 50 are actual frames that were presented to this model in Study 1 of Technical Report 7, which are shown in Figure 51. All of these frames, or *features*, were encoded as hierarchical traces involving the assignment/storage of codes in subsets of L1 and L2 macs, and in the single L3 mac, $M^3_{(0,0)}$. For simplicity, we are assuming here that each of the four afferent L2 macs is frozen and has 500 codes, i.e., 500 L2-scale features, stored in it, for an overall input space (ignoring H inputs) describable as: 500^4 . In fact, the frame active at L0 of the network at right of Figure 50 led to activation in four of 16 L1 macs, all four L2 macs, and $M^3_{(0,0)}$. Looking at this frame's representation in the abstract space (at one end of the double-headed dashed arrow), we can see that exactly four of the 6x6 patches have an appropriate number of active pixels, and so are not shaded. Because each of these four L1 features falls in the RF of a different L2 mac, all four L2 macs become active. However, it is important to realize that the L2 macs are effectively blind to the active pixels in the shaded patches, as is $M^3_{(0,0)}$. Thus, it is specifically the portion of each frame that occurs in non-shaded 6x6 regions that is stored as a feature in $M^3_{(0,0)}$.

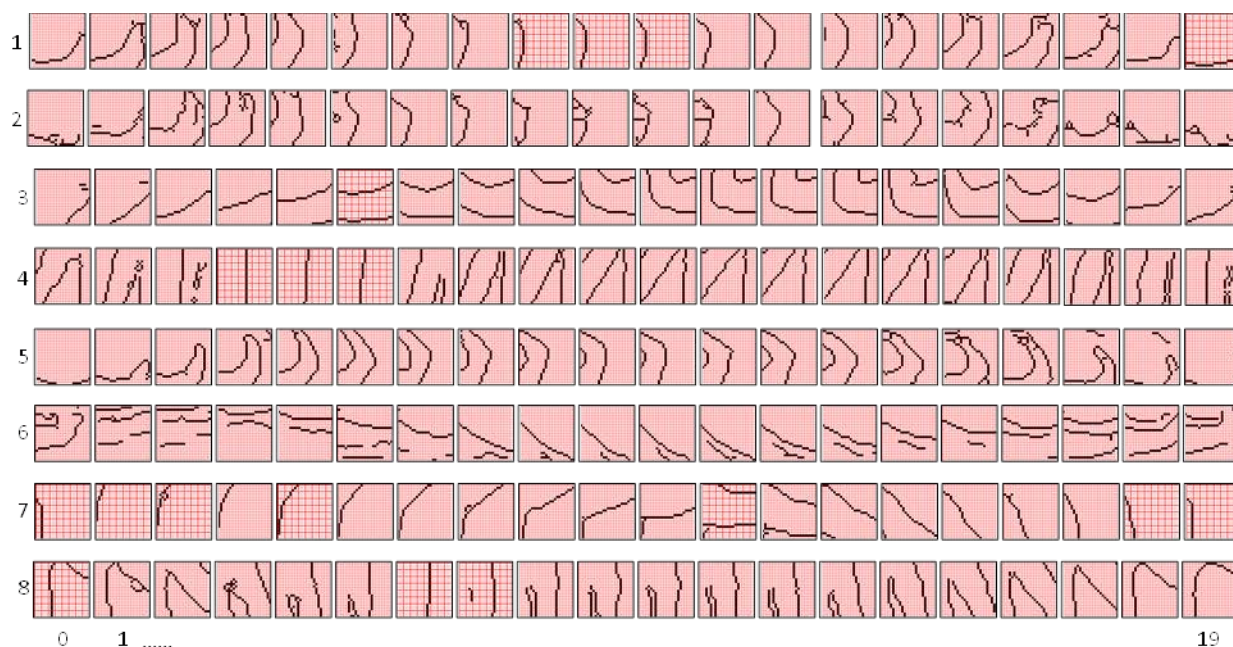


Figure 51: All Frames of the Input Set of Study 1 of Technical Report 7

In Figure 52, we show an alternative view of the network shown in Figure 48 to Figure 50, emphasizing the purely featural view of the activated network across all three internal levels. Rather than show the CMs and cells comprising the macs, we show the abstract input feature/concept space for each mac. The point at the center of a hypersphere corresponds to the feature/concept juxtaposed feature depiction and thus also to the specific SDR code for that feature. The blue lines show the explicit hierarchical connections between the codes (and thus, features) active in this instance; each of these represents the bundles of U and D weights that would be increased between the whole sets of cells comprising the SDR codes at either end of the blue line. Figure 54 is included to make the relation to earlier figures clearer. Figure 53 is included to make the actual features we are discussing clearer than in the 3D views. Figure 52 to Figure 54 emphasize the hierarchical, compositional nature of the representations. The concept active at the top level (L3) is composed of the four L2 concepts, one active in each of the four L2 macs, and each of these L2 concepts (features) just happens to consist of a single L1 concept (feature) active in the L2 mac's RF. In subsequent figures, we will see examples in which the L2 concepts consist of multiple L1 features just as the L3 concept here consists of multiple, i.e., four, L2 features.

Let's take a moment to think about the features at the different hierarchical levels in this example. We might describe the L3 concept (which, from the system's point of view consists of only the four non-shaded patches, and can be seen more clearly in Figure 53) as a "2-sided region that bends slightly and narrows towards the top". We say "2-sided region" because it is not closed at the top or bottom. We external observers of the situation know that this percept is actually part of a person's waving arm—i.e., part of the upper arm, elbow, and forearm. However, given the system's much more limited experience (it's only seen the eight 2-frame sequences in Figure 51), this particular L3 feature must be described as generically as possible.

Hence, our use of simpler concepts like “side”, “region”, “top”, “bottom”, “narrows”, and “bends”, in the description of the L3 concept.

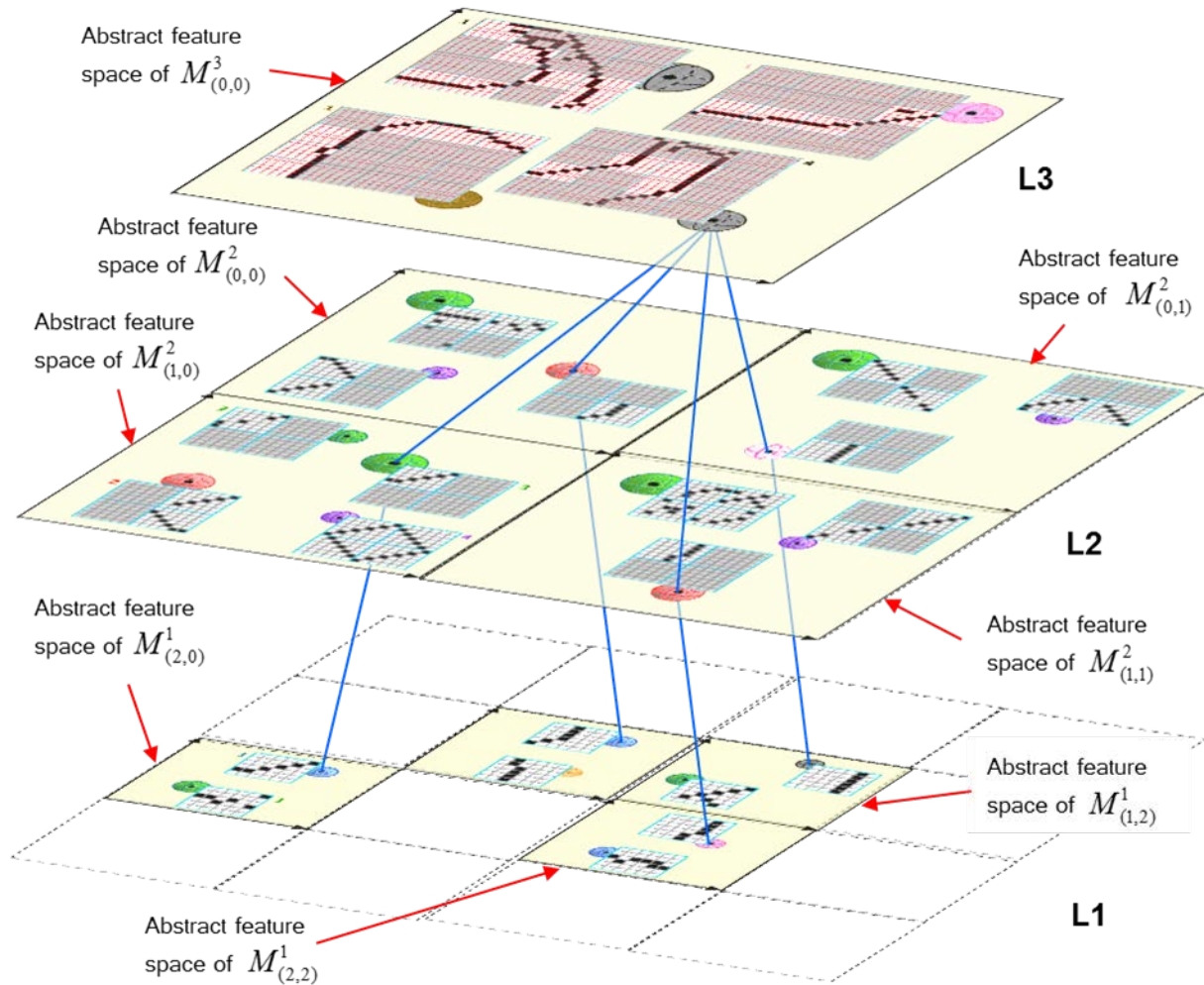


Figure 52: 3D View Emphasizing the Hierarchical, Compositional Nature of the Multi-Level Representations

Now, looking at L2 in Figure 53, we see that these simpler component concepts of the L3 concept correspond to the smaller-scale features active in the individual L2 macs and to spatial relations between those smaller-scale features. That is:

1. In the upper left quadrant of $M^3_{(0,0)}$'s RF—i.e., in the RF of $M^2_{(0,0)}$ —there is a more-vertically-than-horizontally-oriented edge, which also has a “bend” in it.
2. In the lower left quadrant of the L3 mac's RF ($M^2_{(1,0)}$), there is a diagonal edge that is consistent, in two ways, with the “narrows towards the top” feature being true of the higher-scale (L3) concept of which it is a part. First, its orientation ($\sim 225^\circ$) is consistent with “narrows towards the top”, e.g., a 135° edge at that particular spatial location (relative to $M^3_{(0,0)}$'s RF) would not be. Second, its spatial position, both within $M^2_{(1,0)}$'s RF and relative to the spatial position of the active feature in $M^2_{(0,0)}$'s

RF is consistent with an overall “narrowing-towards-the-top” shape feature at the scale of $M^3_{(0,0)}$.

3. In the upper right quadrant of the L3 mac’s RF ($M^2_{(0,1)}$), there is a vertical feature.
4. In the lower right quadrant of the L3 mac’s RF ($M^2_{(1,1)}$), there is a near-vertical feature. The relative spatial positions of these two features are consistent with an also near-vertical edge at the scale of $M^3_{(0,0)}$.

Thus, the component features in the four L2 macs together with the hard-wired spatial relations of those four macs justifies the overall generic description of the L3 concept that we stated earlier, a “2-sided region that bends slightly and narrows towards the top”.

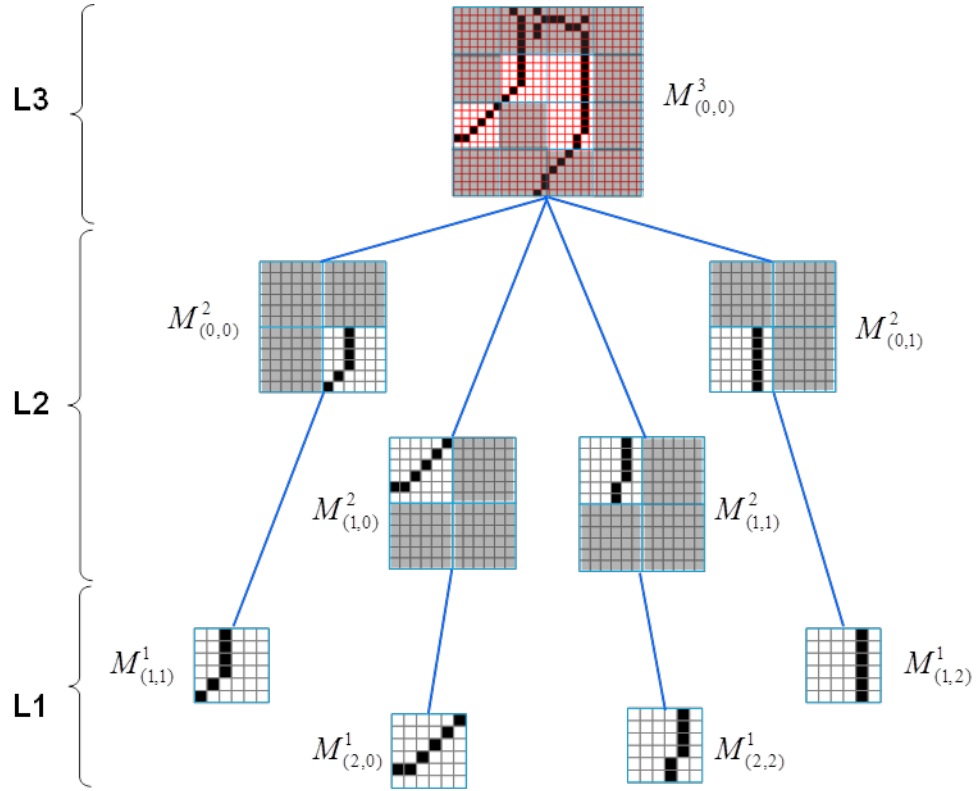


Figure 53: 2D View of Features Comprising the Particular Top-Level Feature Present in this Ongoing Example

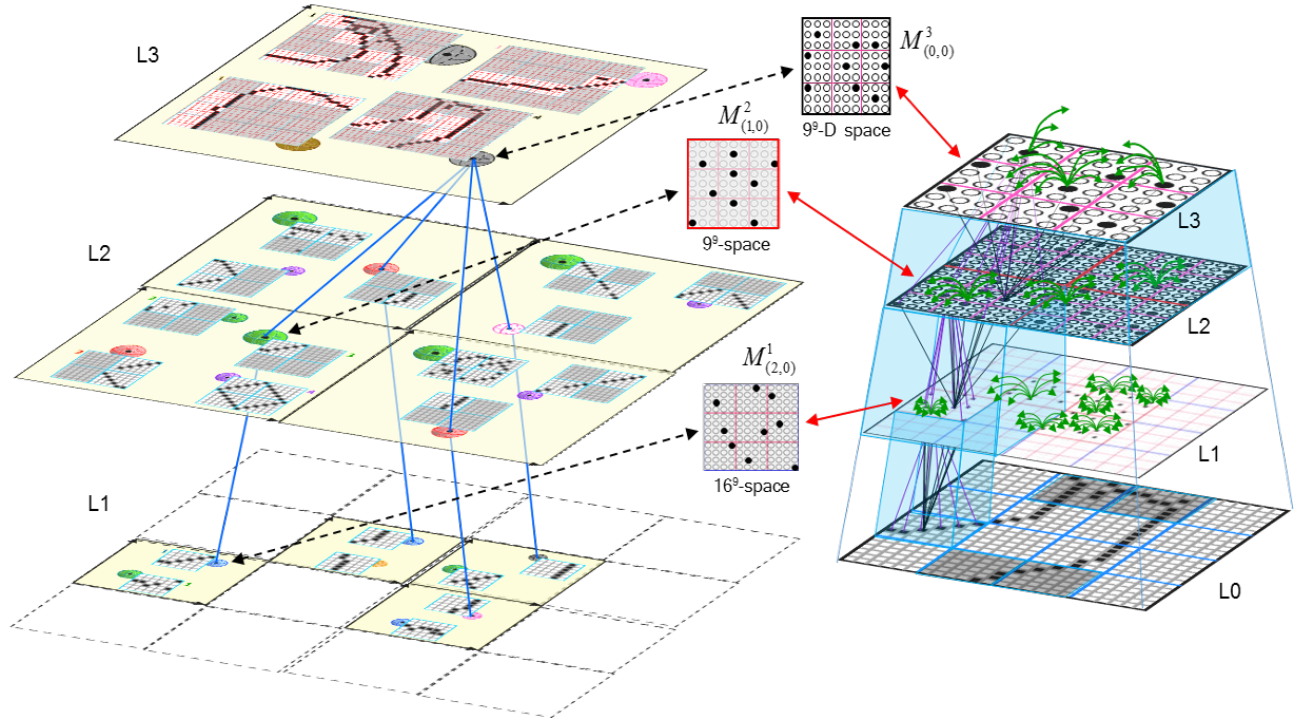


Figure 54: Correspondence of 3D and 2D Visualizations of Compositional Hierarchies

Figure 55 shows examples of four more hierarchical, compositional memory traces of particular moments (frames). In fact, these are the first four frames of Seq. 1 (see Figure 51) presented in Study 1 of Technical Report 7. The format is changed slightly from that of Figure 52 to reduce clutter and shrink the figures: specifically, we show only the single active feature / concept in each mac, rather than a sample of several of the concepts (basis elements) stored in each mac. For this reason, the hypersphere color scheme is essentially irrelevant here. The hierarchical branching patterns (blue lines) make clear that the number of Level J features that compose a Level $J+1$ feature can vary, in this model, from one to four. For example, in panel b, the concepts active in L2 macs, $M^2_{(0,1)}$ and $M^2_{(1,1)}$ each have three composing L1 features, while $M^2_{(1,0)}$ has only one composing feature. In addition, the concept active in $M^2_{(0,1)}$ varies in number of composing features from one frame to the next, from one in panel a, to three in panels b and c, to two in panel d. Figure 56 shows an alternate view of Figure 55c in the style of Figure 53 in order to emphasize the more general case in which the different active macs at any given level, in this case, L2 in particular, can have varying numbers of active composing concepts.

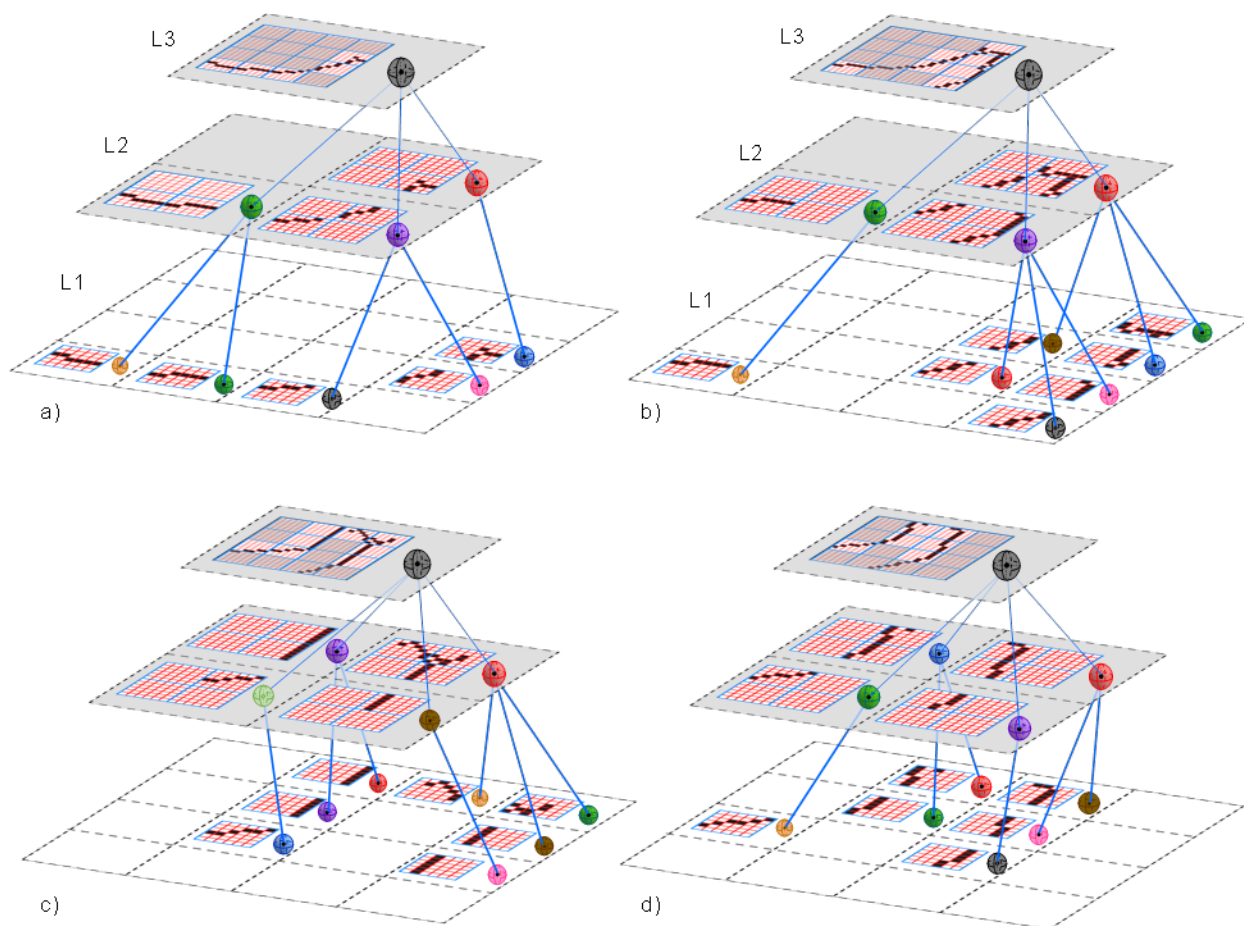


Figure 55: Hierarchical Compositional Memory Traces of First Four Moments of Figure 51

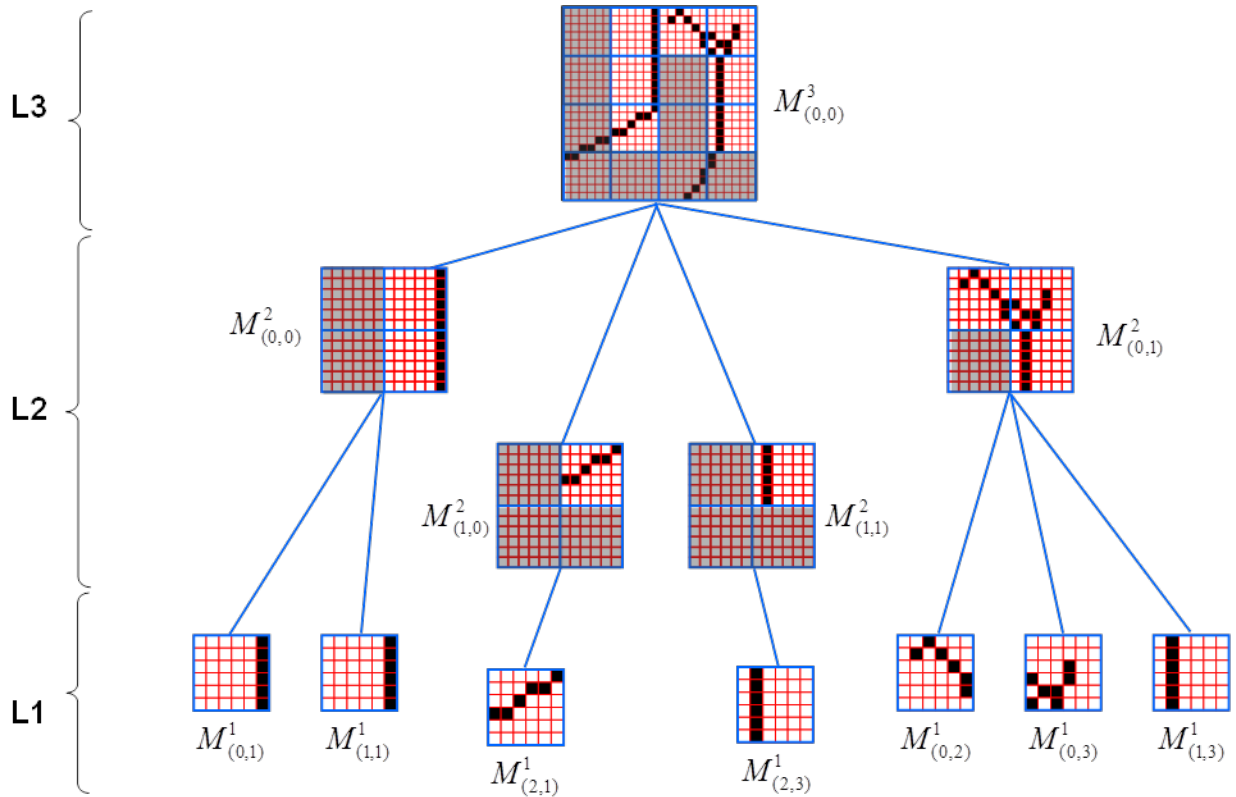


Figure 56: Alternate View of Figure 55c in the Style of Figure 53

The variable numbers of composing features across macs both within and across levels, shown in Figure 55 and Figure 56, underscores the need for a variable normalization process in the model, which has been described in earlier sections. That is, although the number of features comprising the L2 concept (and thus, roughly, the complexity of the L2 concept) varies from one to three, the neurons participating in the codes of these four L2 concepts must register a maximal degree of match in each case. However, their raw U summations will vary widely (in fact, by 3x in this example) across these L2 concepts. Thus, each neuron must know not only its raw U summation, but the *number* of active U inputs (i.e., the number of active composing features) on each frame in order to know how to normalize its U summation. The same is true for the H and D inputs although they are not present in this example.

It may seem that we have belabored the explanation of the hierarchical, compositional nature of the representations here. But, we need this detailed understanding in order to:

- a) Evaluate the adequacy / appropriateness of the features learned by Sparsey, i.e., the features that become elements of the bases of the macs, and
- b) Understand the nature of the invariances learned (and learnable) by Sparsey.

Regarding (a), in the key of Figure 50, we gave a notional set of somewhat more complex features that might be reasonable basis elements over a 24x24-pixel region. These features generally have longer names than those in Figure 49 and Figure 48. And, they are clearly rather

generic. The 24x24-scale feature discussed in Figure 52 to Figure 54 is also generic, though its description is somewhat more complex than those of Figure 50's key and it seems to be more random or ad hoc than those of Figure 50. Let's take a moment to consider that 24x24 feature in relation to the 24x24 feature in Figure 56 and to that in Figure 55a. We show these three 24x24 features across the top of Figure 57. In the dashed box immediately below each figure we show two other very similar instances, which would with high probability cause the same mac codes to activate at all levels, i.e., the same hierarchical SDR memory trace. The key point here is that the inputs to the different L1 macs vary from that in the learned instance, but the variation is globally uncorrelated (i.e., noise-like), and in particular, not of the type that would be caused by structural variation across instances of a category. Such structural variation can be either intrinsic, e.g. a one-armed man is still a man, or imposed by viewing conditions in 3-space, e.g., a person behind a counter looks like they have no legs, but is still a person.

In the next tier down in the figure, we show two other instances of each top-row feature that would be semantically similar to that top-row feature, but which have much wider variation within the composing macs' RFs (at L1 and L2). Due to the large differences between these localized (to individual mac RFs) inputs and those that occurred in the original learning instance, these would lead to very different mac codes in many of the macs at the different model levels, and in particular, in the L3, $M_{(0,0)}^3$. Thus, these second-tier examples show instances in which supervised learning would be needed in order for the model to put these semantically similar, but pixel-wise quite different, inputs into the same categories as the instances at the top.

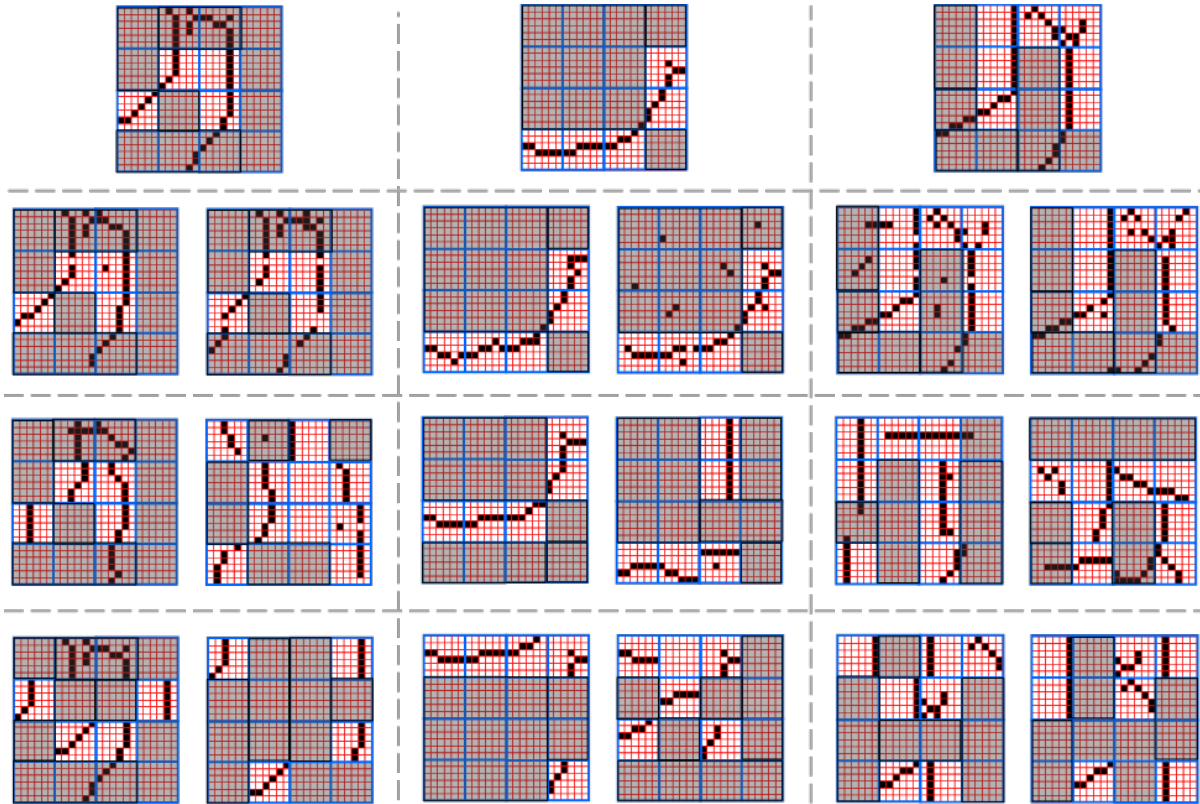


Figure 57: Three Top-Level Feature Instances and Hypothetical Inputs with Varying Pixel-Wise and Semantic Relatedness to Those Instances

In the third tier, we show instances in which the same 6x6-pixel sub-patterns occur, but in the RFs of different L1 macs. That is, the L1 features are spatially scrambled with respect to their spatial arrangements in the instances at the top of the columns. Despite having identical L1 components, these overall 24x24 patterns do not look like plausible instances of the same categories of the patterns at the top of the columns.

One key point of Figure 57 is the notion of invariance. The noise-like variation seen in the first-tier instances of Figure 57 is handled by Sparsey's SISC property. With respect to the bottom-up pattern to each mac at any given level, the noisy instance would correspond to a point falling within one of the hyperspheres in the mac's high-dimensional abstract input space. The variation seen in the second tier examples, which is more indicative of structural changes in the input space (either intrinsic or due to 3D viewing conditions), correspond to highly nonlinear transformations (from the original learned instance) and require a supervised learning process to be learned. The variation in the third tier examples is also consistent with (gross) structural change, but these examples are not likely exemplars of the original instances (tops of the columns) from which they were made. This emphasizes that strong spatial constraints on the classes of objects are present in natural worlds. These constraints are to a large extent automatically adhered to by virtue of explicit spatial arrangement of the macs at each level of the visual processing hierarchy. In other words, these constraints do not need to be explicitly computed, either during learning or retrieval, which saves a huge amount of computational costs relative to a system in which these spatial constraints are not reified in architecture.

A second point visible in Figure 57 and in all of the prior figures as well is that in our approach, the features/concepts—i.e., the basis vectors stored in any given mac—can generally look quite random or noisy. Thus, Sparsey's approach to pattern recognition differs fundamentally from that of the vast majority of existing pattern recognition, i.e., classification, models, which use hand-designed features from the lowest level to the highest. That is, recognition systems often have higher level models of objects and/or events, often expressed in some (probabilistic) grammar, or other formalism, which are used to constrain the bottom-up recognition process (e.g., the "language" model in speech recognition systems). Sparsey does not use or require any such higher-level a priori knowledge.

The features that macs store are actually spatiotemporal, not purely spatial.

All of the foregoing in this section has been explained in terms of purely spatial features / concepts. However, Sparsey is an inherently spatiotemporal model and has been explained in those terms in all of the prior technical reports of this research program. Thus, each code stored in a mac actually represents a particular temporal context-dependent *moment*, not simply the purely spatial input presented on a particular frame. Depicting this added complexity in the format of the prior figures is difficult, and so for simplicity, the foregoing exposition and figures pretended as though the features being stored were purely spatial. But for completeness and to accurately convey what's being stored in the model, Figure 58 shows that the features corresponding to each hypersphere are actually moments, i.e., frames in the context of the longest unbroken frame sequences leading up to them. Thus, Figure 58 generalizes Figure 48. We will develop similar figures for higher levels in subsequent work. Figure 48 shows that the SDR codes stored in the L1 macs actually represent moments of varying duration but of the 6x6-

pixel spatial scale. Note that formally, we have changed the abstract feature space from being 36-dimensional to being $36T$ -dimensional, where T is the number of prior frames of context. In Sparsey, T is not of fixed order.

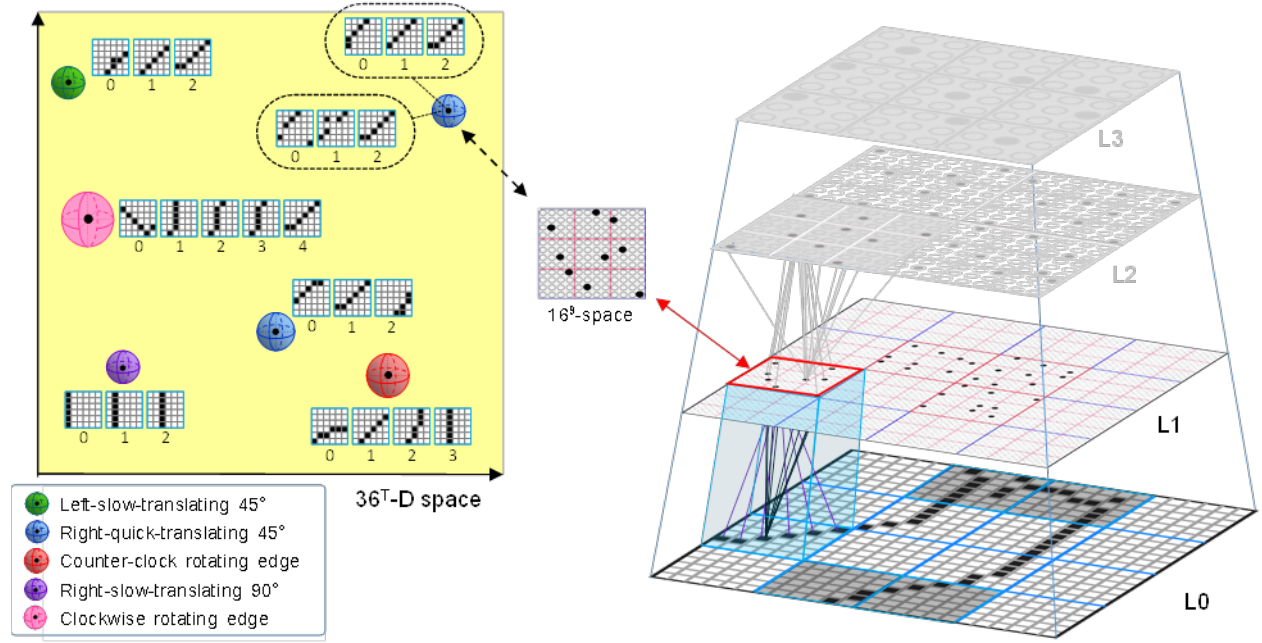


Figure 58: Spatiotemporal Generalization of Figure 48 Showing that SDR Codes Actually Represent Spatiotemporal Features

6.4 Optimal Normalization Thresholds

Figure 59 shows a situation in which a higher U normalization threshold would yield substantially higher correctness of the reinstated code. Determining the optimal normalization thresholds (and policies) as well as optimal weight saturation thresholds for the U , H , and D different matrices for all levels will remain an ongoing pursuit in our follow-on research, one which will be carried out simultaneously with optimization of the U -RF, π bound, and other parameters. Read the figure annotations in numerical order.

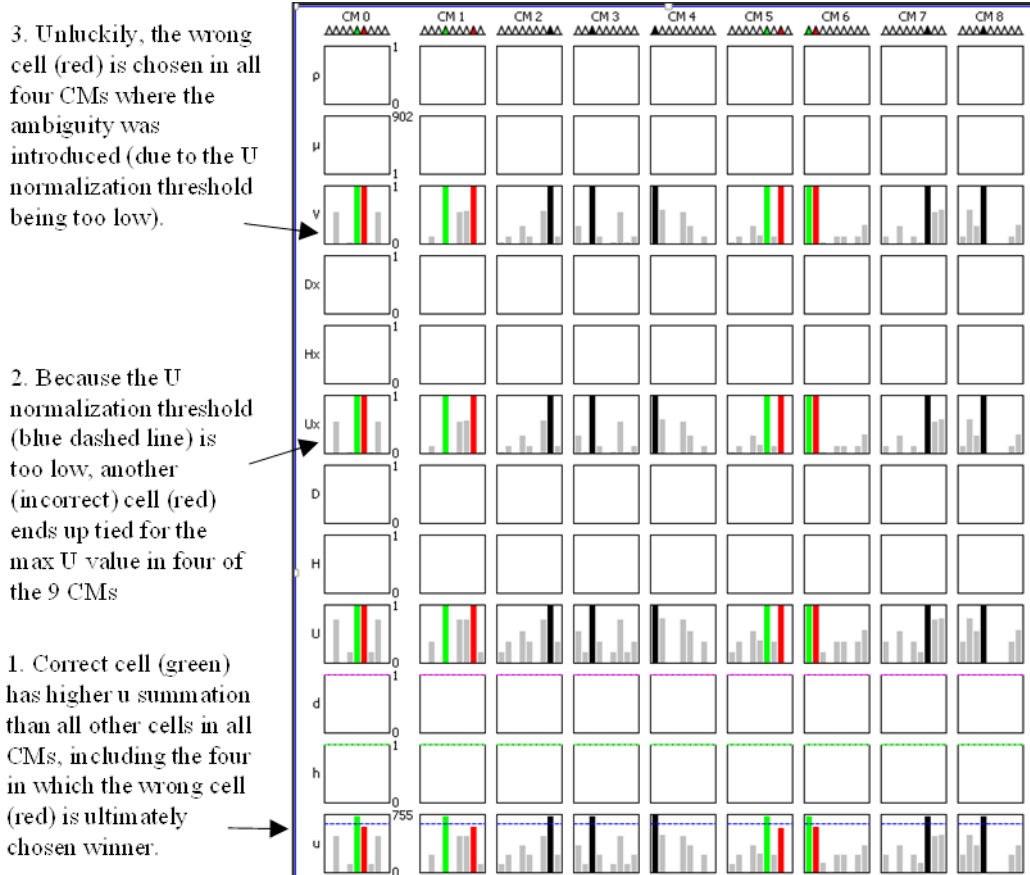


Figure 59: A Case in which a Higher U Normalization Threshold Would Yield Substantially Higher Correctness of the Reinstated Code

6.5 Fraction of Represented Features Should Remain near 100% at All Levels

One of our main research goals was to understand how representations of parts, i.e., ecologically meaningful parts (segments) of objects/events, automatically and unsupervisedly emerge during learning in Sparsey. In Appendix A of TR16, we introduced what can be viewed as a prerequisite capability that must be present if the model is to be able to automatically learn to recognize natural parts of objects and events. Specifically, the model's parameters must be set so that all or most of the input features (i.e., the active pixels in the input level, L0) end up *being represented* at the model's top level. By "being represented" we mean "influencing the choice of cells comprising the code". Being represented at the top level entails being represented at all the intervening levels as well.

Figure 60 clarifies this issue. It shows the active macs (rose shaded) at all levels on each of four frames while processing an 8-frame snippet. Superimposed on each level, it also shows the portion of the input pixels *represented* at that level. One can see that on all frames, a substantial fraction of the input pixels in all regions of L0 are represented at L1 and L2. However, one can also see that on frame 2, the entire left half of the input pattern (which shows a part of an arm) drops out of the representation at L3 (green arrows indicate the comparison). Why do they drop out at L3? As explained in TR16 Appendix A, it is due to the interactions of the sizes of the

U-RFs and the mac activation bounds (π^- and π^+) for macs comprising the left side of L3. In contrast, the combination of U-RF, π^- , and π^+ , settings for the three active L3 macs happen to be appropriate for them to be activated. Consequently, a significant fraction of the pixels in the right half of the input image end up being represented at L3. Similarly, the parameters of the rose-shaded L4 mac are also appropriate for it to become active. Note that L5 does not become active on frame 2 because we have introduced an explicit level-specific hard delay parameter (referred to as “staged activation” in TR16) to the model and in this simulation, L5 could activate no earlier than frame 3.

In any case, the situation at frame 2, in which an entire macroscopic region of the input (e.g., the left half) drops out of the representation at higher levels must be avoided. If mid or higher level object/event classes are generally defined in terms of their parts and the relations between their parts, then we must ensure (at least statistically) that all parts of the input image remain represented at the higher levels. Thus, finding general rules for the probabilistic assignment of parameters, which maximize the fraction of input features (pixels) represented at the top (and all intervening) levels is a key goal. In fact, one can see that on frames 5, 6, and 7, the overall input image is much better represented up through all levels. Even though perhaps only 40-50% of the pixels remain represented (survive) at L5 on frames 6 and 7, the particular pixels that do survive reflect the overall shape of the input image. Nevertheless, we found that such low percentages of represented features often allowed well-structured elements of the input image to fail to be registered at L1.

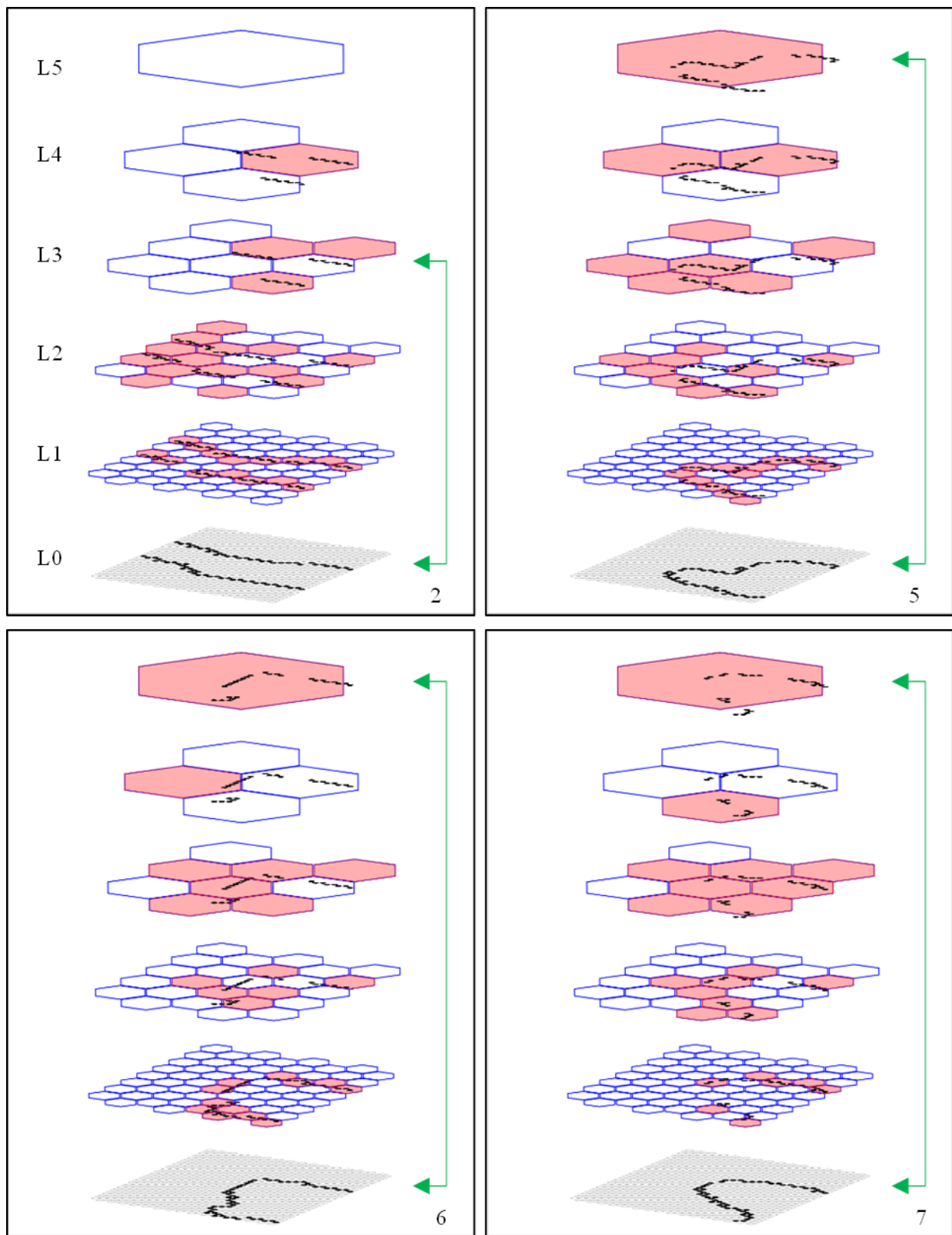


Figure 60: Surviving Input Features at All Internal Levels on Four Example Frames

Around the timeframe of TR16, we generalized the model’s wiring policy to allow the U-RFs of the L1 macs, which are regions of pixels, to overlap. This greatly increases the chances of any given active pixel falling within the U-RF of at least one L1 mac that meets its $[\pi^-, \pi^+]$ activation criteria. Figure 61 shows how L1 U-RFs now overlap. (Actually, this kind of overlap can also be seen in many earlier figures in this report, e.g., Figure 3, Figure 19, and Figure 20.) The blue line sprays show the U-RFs of two neighboring active L1 macs (A, B) and the red arrows indicate two pixels shared by both U-RFs. These two pixels influence the codes chosen in both macs. There are five active pixels in A’s U-RF and six in B’s, which must meet their respective π criteria (or they would not be active). Thus, each of these two pixels (features) has two chances to contribute to an overall local activation pattern that satisfies criteria and thus be encoded at L1.

Allowing LI U-RF overlap led to an immediate large increase in the fractions of input pixels represented throughout all model levels. The models used in many of our studies attain 90-95% effective feature representation. In follow-on work, we will make modifications to bring that to 100%. This same principle whereby overlapping RFs (U, H, and D) maximize the fraction of the input pixels (features) represented by active mac codes operates at all model levels.

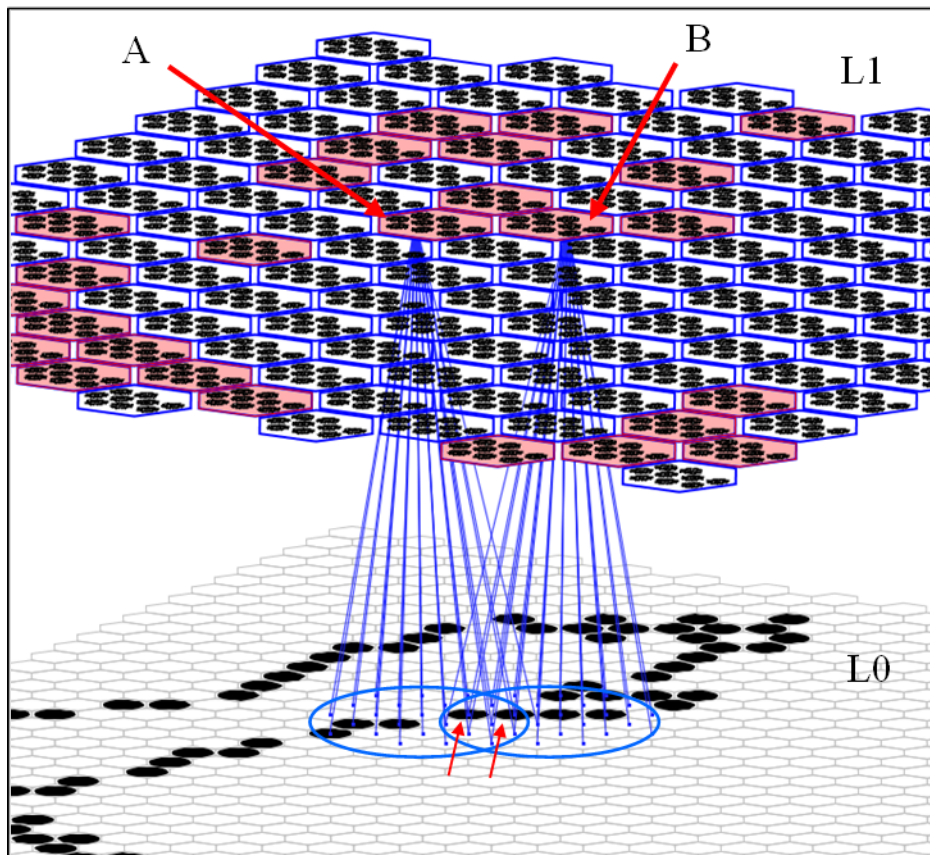


Figure 61: Illustration of Overlapping L1 U-RFs

6.6 Hierarchical Compression

It might seem counter-intuitive to have the goal be that *all* input features remain represented at the highest level. Typically, one thinks of the highest levels of a representational hierarchy as having a more abstract/summary character, i.e., omitting much of the lower-level details. Amongst other things, this would seem essential in order for a high-level representation, i.e., code, to represent invariances. We make the following points in answer to this concern.

1. Even if the information from *all* parts of the input image reaches the top level and thus influences the selection of the top level code, it is nevertheless the case that that top level code will generally consist of a far smaller number of active cells than lower level codes. In general, the number of active cells falls sharply with level. Figure 62 illustrates this progressive code compression, in particular for frame 5 of our example. The represented (coded) information is at left and the specific codes (black or black/gray cells) and number of active cells per level is at right. Moreover, the increasing code persistence at higher levels means there is increasing temporal compression as well. In this particular simulation for example, L4 (L5) codes persisted for 4 (5) frames, respectively, meaning that for a 5-frame span with approximately the same amount of input activity on each frame, a single 9-cell L5 code can represent approximately the same information as order 500+ L1 cells, for at least 50x compression.
2. Even if all input features influence the choice of cells comprising the L5 code, that does mean that those features are *explicitly* present in that code per se. That is, merely knowing the set of $Q=9$ cells comprising the L5 code in Figure 62 does not recover for us the details of those features. For that, we would have to propagate D (and H) signals, down through the hierarchy and across time to cause the entire hierarchical trace to recapitulate. We've generally referred to this process as retrieval of a stored memory but it can also be viewed as un-packing or decompressing the top (and intermediate) level codes.
3. Regarding the abstractness issue, note that in general, different spatiotemporal input patterns, i.e., different spatiotemporal *feature* patterns, can cause the same L5 code to activate. In that case, said L5 code would implicitly represent whatever invariances exist over the set of inputs that cause it to activate. Other input patterns will cause other codes to activate. It is the pattern of overlaps over the set of codes stored that encodes the higher level statistical (i.e., abstract) structure of the input space. Thus, our stated goal that all input features should ideally be represented (i.e., influence code selection) at all levels does not preclude higher level codes from being abstract and capturing certain higher-order statistical regularities (invariances) in the input space.

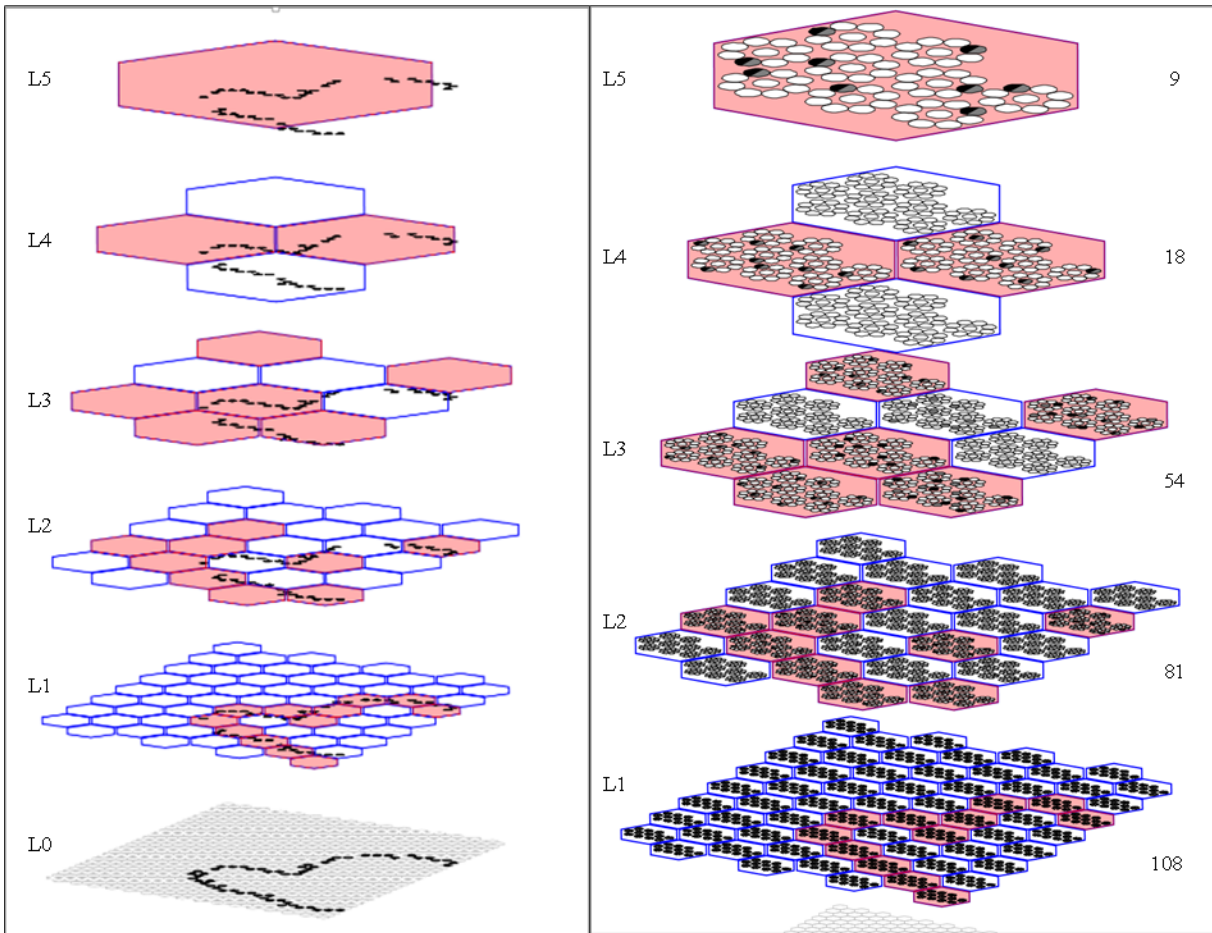


Figure 62: Illustration of Progressive Code Compression up Through the Model's Levels

6.7 Trace Accuracy can be Quite Low, While Supporting High Classification Accuracy

One of the most robust properties that we are seeing in our multi-level (~5-8 levels) simulations is that even though recognition accuracy at the lowest levels, e.g., L1 and L2, might be low, e.g., 60%, recognition accuracy at the higher levels can be nearly 100%. At first glance, this seems counterintuitive: how can higher level processing, which receives signals from lower level processes, do better than the lower level processes? Thus far, we can see two principles responsible for this property:

1. Recall a mac at level J often receives input from multiple level J-1 macs. The codes active in those level J-1 macs may individually have errors. However, if those errors are not correlated, then when the U signals arriving from those macs are added together, the resulting summed U input distribution to the cells of a CM may still be peaked at the correct cell (the uncorrelated errors having averaged out). Figure 63 shows a specific example of this.
2. Recall that in Step 4 of Sparsey's CSA, the three input signals to each cell in a mac, the U, H, and D signals, are multiplied to yield V. And recall that during recognition, we simply pick the max V cell in a CM as winner. The H and D signals carry temporal context information and in particular, the D signals carry larger-scale (i.e., more slowly varying) temporal context

information because they originate from higher level cells which generally have longer persistence. We observe quite frequently that while the individual U, H, and D vectors of signals to the K cells comprising a CM have errors (including the case where multiple cells are tied for the maximal value), the product of the three vectors, the V vector, is peaked at the correct cell. Again, the errors in the three factor vectors, being uncorrelated, are attenuated due to the multiplication process. This effect can also be seen in the example in Figure 63.

We believe that this finding may be quite important in understanding the design principles of hierarchical information processing systems and anticipate generating many related results in the coming months. This finding also raises quite interesting issues related to episodic memory. Specifically, it suggests that when a person engages in detailed episodic recall of specific experienced events, while the memory trace playing out in the higher levels might indeed be highly veridical to the original experience, the unfolding traces at the lower levels—i.e., the spatiotemporal pattern of activations across hundreds of L1 and L2 macs over many frames—might actually contain a substantial fraction of errors (i.e., incorrect winners in CMs). The experience of recalling a specific event might feel subjectively highly accurate partly if one's conscious awareness/attention depends more on the higher level macs than the lower level macs. There have been many studies (e.g., concerning the veracity of eyewitness testimony) showing that the actual low-level details of remembered events are often wrong, essentially having been filled in by statistically (semantically) reasonable lower-scale component events (i.e., confabulation). In the terminology we have used in the last several reports, such a confabulation event would be the reactivation of a closely matching, but not the exactly correct, basis element stored in the mac. In fact, if the original event occurred *after* L1 froze, then the original encoding of the event would have been in terms of already-learned (closest-matching) basis elements (i.e., even during the original perception of the event, the raw sensory inputs would already be channeled into familiar vocabulary).

As discussed above, it has become clear that the accuracy of information retrieval processes generally (i.e., for most parameter setting tested, though again, we are far from having a complete understanding of the huge parameter space) increases with level and that even when L1 and L2 accuracy is in the 40-50% range, accuracies in the 80-90% are realized at the highest network levels. Figure 63 shows an example of this situation in which the two L1 macs, M_{302}^1 and M_{284}^1 (shaded purple), providing U input to an L2 mac, M_{166}^2 , have errors in their codes, with 43% and 71% accuracy, respectively, yet the code in M_{166}^2 is 100% accurate. Note that due to size limitations, the individual cells/CMs of the macs are not shown here. In the plots, black/red/green bars indicate correct winners, incorrect winners, and incorrect non-winners, respectively.

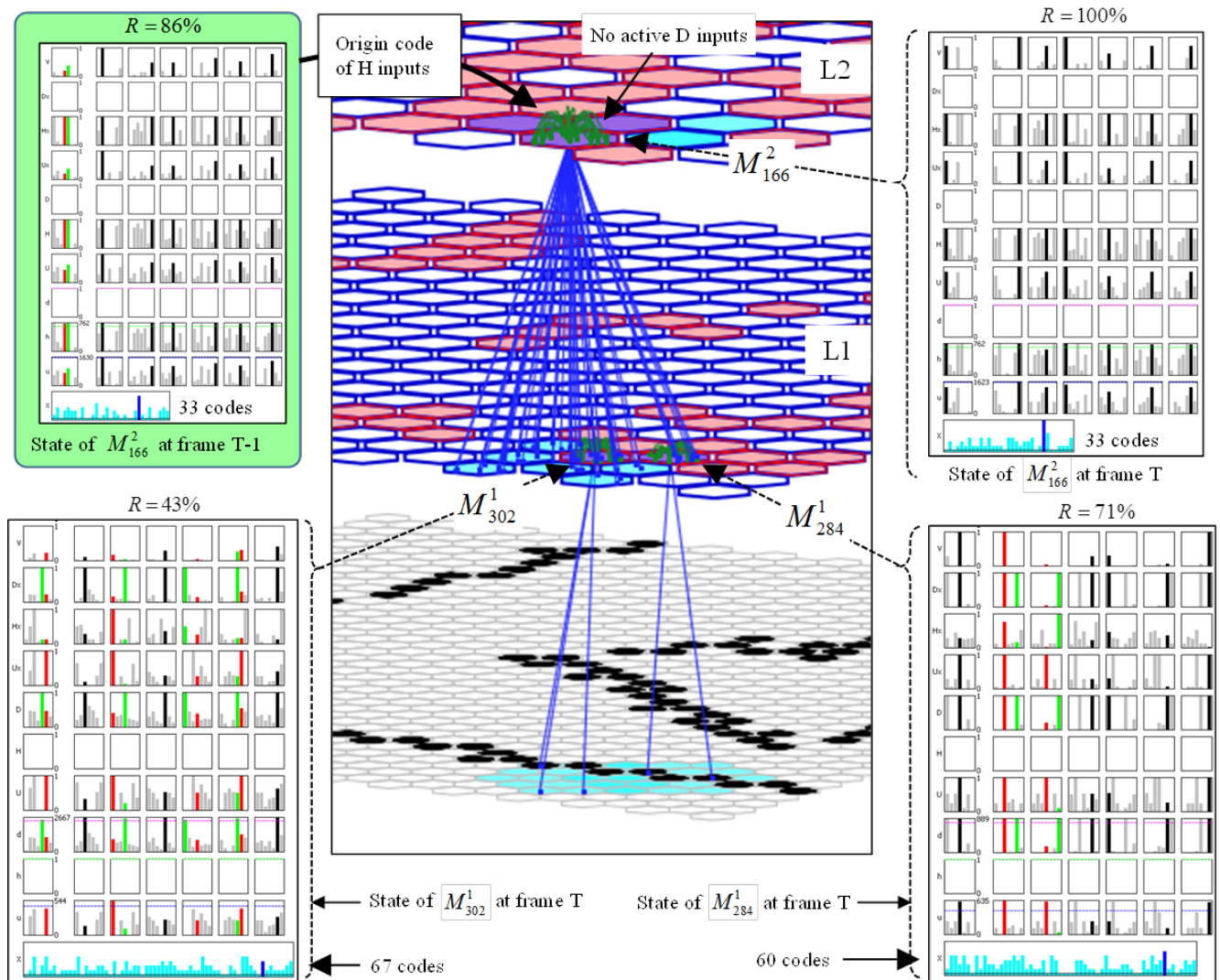


Figure 63: Example of Highly accurate Recognition at Higher Level Despite Highly Inaccurate Recognition at Lower Level

Note that this figure zooms in on what is happening only in a tiny section of the 7-level 712-mac network on one particular frame of one of the 45 8-frame snippets presented (a single time each) during the learning phase of one experiment. This type of situation in which the combination of various sources of information via both the additive and multiplicative principles specified in the introduction (bullets 1 and 2) corrects errors and either maintains or improves the accuracy of the unfolding spatiotemporal hierarchical memory trace typically occurs hundreds/thousands of times during the recognition of even a single 8-frame snippet.

At upper left of Figure 63, we show the conditions that existed at the time the prior code (at T-1) was chosen in M^2_{166} . That code was chosen as the product of U and H signals and one can see (you may have to zoom in a bit) that in six of the CMs, the product of the filtered U and H signals, denoted “Ux” and “Hx”, result in the correct cell (black bars) having the maximal V value, yielding an 86% correct code. Signals from that code arrive via the recurrent H wts (green) at time T and are combined multiplicatively with even more error-full U signals from the

two L1 macs, M_{302}^1 and M_{284}^1 . At upper right of the figure, we see these various incoming signal vectors to M_{166}^2 at time T. We point out that although the H signals are more ambiguous (i.e., there are 2-3 cells tied for the maximal Hx in each CM), multiplication with the more selective U signals leaves only the correct cell in each CM with a maximal V value.

To summarize this example, we have a case where three sources of information have accuracies of 86%, 43%, and 71%, respectively, and combine via the CSA to retrieve (activate) a code with 100% accuracy. Note that in this particular case, there were no active D signals to this L2 mac. While we as yet have a great deal to learn about these dynamics, we emphasize that these “evidence combination” dynamics are apparently largely working correctly in the context of quite sizable models and furthermore, working in the fully spatiotemporal case.

6.8 Minimize the Number of Post-Quiescent Mac Activations

In general, we will want H-RFs to extend to at least some neighboring macs. This is because doing so increases the number of instances in which a mac becomes active *in a context*. Perhaps the most difficult general problem in sequence recognition is the problem of having to distinguish large numbers of sequences that start with the same item (or sequence of items), call it the “common prefix” problem. This was the motivation for inventing the Overcoding-and-Paring (OP) method. Note that this potential problem applies on an individual basis to each mac in a network (since the macs operate autonomously).

As the size of the input surface (L0) and L1 sheet of macs grows, the number of instances in which an L1 mac is active on a frame T after having been inactive at T-1 can become a substantial fraction of all L1 activations. Let’s call such an activation event as a *post-quiescent activation* (PQA). We distinguish this from instances in which a mac is active at T but was also active at T-1, which we will call a *post-active activation* (PAA).

Figure 64 shows one of the 8-frame snippets with the 8x8 grid of 4x4-pixel L0 apertures indicated. All PAAs are highlighted in yellow. The rest (unhighlighted) are therefore PQAs. At L1, PQAs are clearly the rule rather than the exception, occurring about 70% of the time, while at progressively higher levels, the PQA/PAA ratio drops.

If H-RFs include only source macs—i.e., if a mac’s neurons receive H-wts only from the other neurons in their own mac—then in all PQAs, the code is chosen only on the basis of the pattern in a mac’s U-RF (disregarding the D-RF for the moment). This is true even though most of these PQAs will occur on non-initial frames of sequences, and therefore on frames on which temporal context information would often be present if H-RFs included larger neighboring regions of macs. The problem with macs choosing codes based only on U-signals is that the space of possible U-signals is exponentially smaller than the space of UxH-signals, and therefore crosstalk effects (which degrade storage capacity and recognition accuracy) will accumulate much faster. We therefore would like to minimize the number of instances in which mac codes are chosen based only on U-signals. Consequently, we will focus on parameter regimes in which the H-RFs of macs at all levels include at least some surrounding macs.

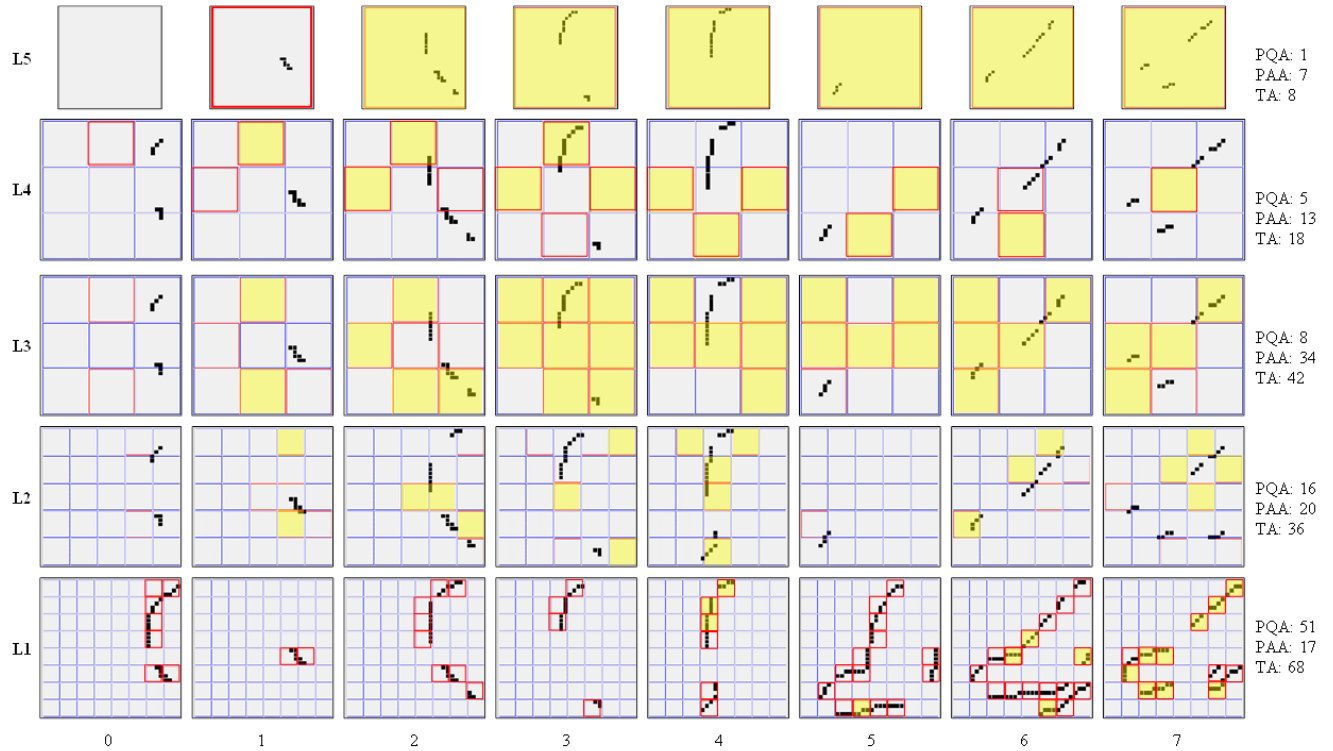


Figure 64: Demonstration of the Large Fraction of All L1 Activations that are PQAs

6.9 Correct Cells are Correlated, Incorrect Cells are Not

One major finding of our research has been an essential principle by which Sparsey is able to learn invariances:

When many macs are involved in the class decision, the codes in those macs can contain many errors (i.e., low trace accuracy) while still supporting high class accuracy. This is because the correct cells across all active macs influencing the class decision are highly correlated with the correct class cell, whereas the incorrect cells across those macs are far less correlated. Thus, the input summations of the correct class cells will tend to rise above the summations of all the other (incorrect) class cells.

However, there is an even more subtle principle that boosts Sparsey’s recognition power. When multiple macs contribute to the summations in a given target mac, the codes in those afferent macs can be even completely erroneous, while still enabling the target mac to activate a significant fraction of the correct cells. This is because even though all cells in all active afferent macs may be wrong, those afferent macs will still have enforced SISC during learning. This means that there will still be positive, though probably small, correlations (through second-order effects) amongst the wrong cells across all afferent macs.

An instance of this principle is shown in Figure 65. It shows the state of the model on the last frame (frame 14) of the “Daria-bend” snippet of the Weizmann dataset. The model has two internal levels, L1 and L2, with 18x26 and 12x16 macs, respectively. L2 mac Y mac has 5 CMs

and only two of the winners are correct (black, discernable in insets A and C). In the other three CMs, the winner is incorrect (red) and the cell that should have won is green. Inset C uses the same color scheme and shows the detailed levels of most of the algorithm's intermediate variables: gray bars are for the other cells in the CM. L2 mac Y has only two active afferent L1 macs at this moment (B and C) and no H or D input. Therefore the code selected in mac Y depends only on the U-input from these two L1 macs. As can be seen, both macs' codes are entirely wrong (red cells, which correspond to red bars in insets D and E).

The reason why L2 mac Y is able to recover the correct cell in two of its five CMs even though all 12 of its input source cells (six in each of the two L1 macs) are wrong can be stated as follows:

1. Let Ψ_1 and Ψ_2 be two similar learned moments.
2. Suppose we are now at Ψ_2 of a test.
3. Because of the similarity, the set of L1 macs activated in Ψ_1 will have a substantial intersection with the set of L1 macs activated in Ψ_2 .
4. Also because of the similarity and because of SISC, the codes active in any such common L1 mac in the two instances will have a substantial intersection.
5. By points 3 and 4, set of L2 macs activated in Ψ_1 will have a substantial intersection with the set of L2 macs activated in Ψ_2 .
6. By points 3, 4, and 5, the codes active in any such common L2 mac in the two instances will have a substantial intersection.
7. Because SISC is a basic property of a mac, it follows that for any L1 mac that is active in both the prior moment and current moment, the expected intersection of the codes active in those two instances should be higher than if the two moments in question had lower similarity. Denote the class of L1 cells in that intersection, ξ_1 .
8. The effect of any cells in ξ_1 on any L2 mac that is efferent to any such L1 mac, or any set of such L1 macs, will be to cause the expected intersection of the code activated in such an L2 mac on the prior moment and the code activated in it on the current moment to be higher than if the two moments in question were less similar. Denote the class of L2 cells in that intersection, ξ_2 .
9. Weight increases that were made between cells in ξ_1 and cells in ξ_2 on that prior moment will contribute disproportionately to the summation of cells ξ_2 .
10. Since cells in ξ_2 are by definition common to the code active in the L2 mac in the prior similar moment and in the code active in the training moment corresponding to the current test moment, such cells are correct.

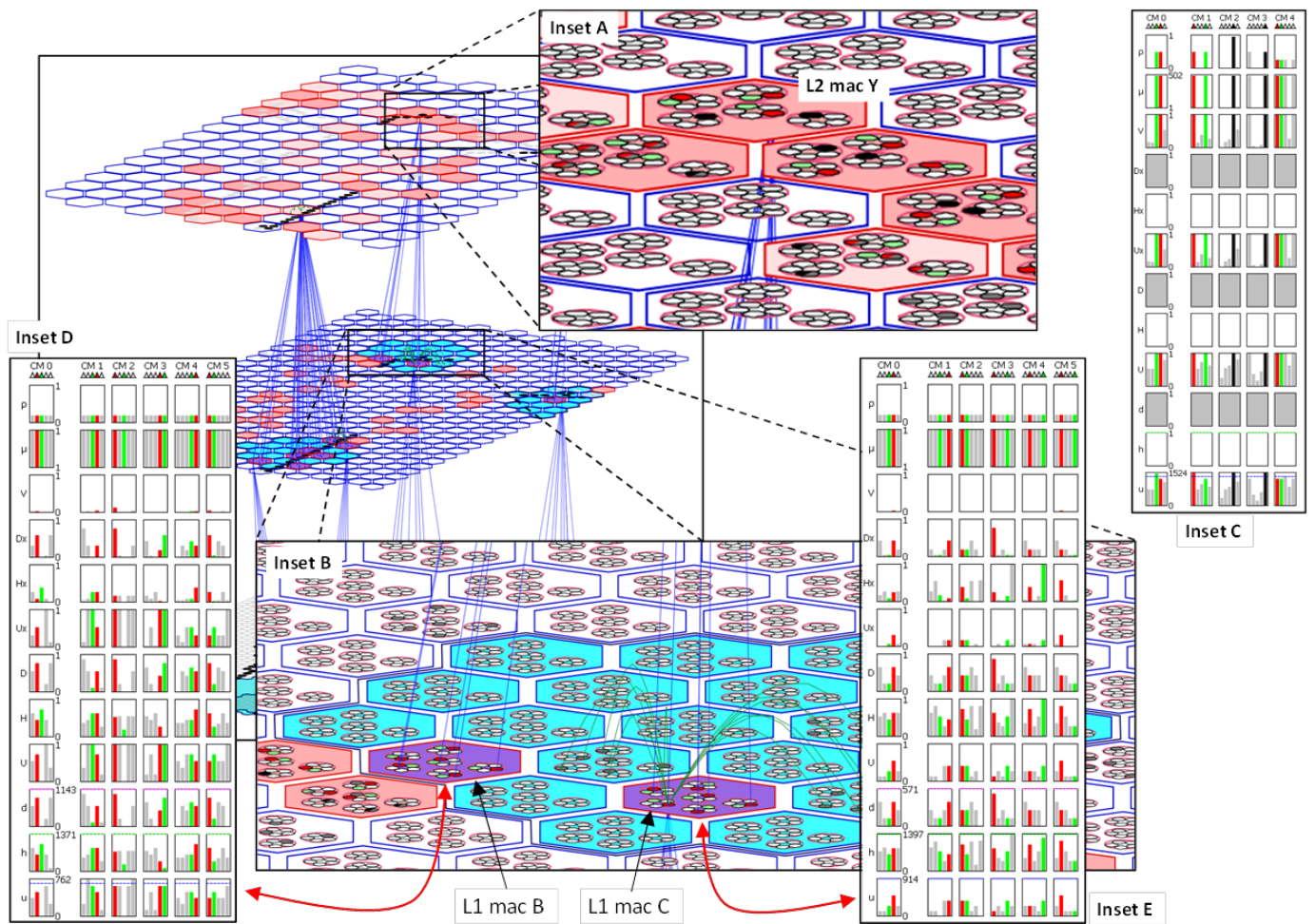


Figure 65: Example of Partial Recovery of Code Accuracy Despite Completely Incorrect Inputs Due to Second Order Correlations Resulting from SISC Property

7. CONCLUSIONS

The biological brain, and the human brain in particular, remains the most powerful information processing device known. The remarkable structural homogeneity across the entire neocortical sheet suggests a core computational module, i.e., a “canonical cortical microcircuit”, operating similarly in all regions (Douglas, Martin et al. 1989, Douglas and Martin 2004, Rinkus 2010). The overarching rationale for our research is therefore that if we want to build computers that process information as well as humans, then we should understand the detailed structure and operation of said canonical cortical algorithm/circuit. Based on a large body of evidence, we identify the canonical module with the cortical “macrocolumn” (“mac”) (a.k.a. “hypercolumn” in early visual cortex, or “barrel”-related volumes in rat/mouse primary somatosensory cortex). Also based on a large and increasing body of evidence, we believe the two most essential principles of intelligence are: a) representing information with SDRs; and b) hierarchical, or *heterarchical*, organization of the overall knowledge base, i.e., in part-whole, fashion in which parts at any level may components of many higher-level wholes.

The overall goal of our two-year project was to investigate how representing information in the form of hierarchical sparse distributed representations SDRs yields extremely efficient solutions to spatiotemporal recognition, and more specifically, video event recognition, problems. The most important result of our research was the continued confirmation and elaboration of the facts that both learning and best-match retrieval of spatiotemporal patterns, result specifically from the use of SDRs in all representational fields at all levels of the heterarchy. As this is highly uncharted territory, this has been very much a basic research project. In addition to this primary result, we have greatly increased our knowledge of many other essential principles of intelligent processing, particularly in regard to the compositional nature of the overall, ongoing computations and how they give rise to invariance.

In the latter half of the project, we turned our focus more towards the applied research goal of achieving SOA classification performance on benchmark video event recognition problems. We have carried out studies involving the Hollywood 2, KTH, and Weizmann data sets. In the final months of the performance period, we have focused almost exclusively on Weizmann in our effort to attain SOA performance. As of this date, the best performance of the specific model we are working with, Sparsey[®], is at 67% which is significantly below SOA, which is 100% (chance = 10%). But, for the reasons stated at the outset of this report, we believe we will achieve SOA classification with even substantially faster times in the very near future. We emphasize that these are the first results we know of a model that uses SDR at all internal coding fields performing on any benchmark video recognition problem. Furthermore, this is the first published result of a hierarchical model based on SDR performing any benchmark task.

Given the model's extremely simple representation of unit state and weights, which implies that very low precision is sufficient, and that fact that its operations, learning and retrieval, have constant time regardless of how much information is stored, we believe that Sparsey will be a strong platform on which to base extremely low-power, extremely fast, and extremely scalable applications in the realm of spatiotemporal (and as a special case, spatial) pattern recognition, going forward. This includes any type of purely sequential pattern recognition task, e.g., any type of language processing application, as well.

We have also only begun to scratch the surface of leveraging *transfer* learning. Clearly, all natural video involving human-centric actions share the same low-level features, e.g., oriented edges, oriented moving (translating + rotating) edges, and other regular low-statistical order features. We should therefore expect that once these low- and perhaps mid-level features are learned, probably on a relatively small amount of data, all subsequent high-level video tasks will be able to proceed much more quickly. As we are still working out the basic model principles and parameters needed to learn our first data set (Weizmann), we are not yet able to demonstrate the potentially large amortization of learning (reduced learning time across data sets/tasks) that will be possible due to transfer learning.

7.1 Importance of Unitary Explanation of Episodic and Semantic Memory

Human memory can be broken into two types: a) *episodic memory*, which is highly detailed memory for specific, experienced events, which can last on the order of a lifetime; and b) *semantic memory*, which is knowledge of classes of objects/events or more generally, of the statistical/causal structure of the world. Almost all (object, speech, video) pattern recognition work to date, including the Deep Learning thread, implicitly concerns only semantic memory. The paradigm has been largely one of batch learning in which class boundaries are gradually learned (generally using gradient-based methods) so that new exemplars can be correctly classified, but there is almost never any demonstration that such systems can also output detailed replicas of any of the individual exemplars used to train them. The details of individual training experiences are lost as the model becomes an optimal classifier.¹¹

However: a) human cognition clearly includes often highly capacious and long-lived episodic memory (and by definition, episodic memories are of events that occur only once, i.e., single-trial learning); and b) it is possible that a single architecture/algorithm can accomplish both episodic and semantic memory. Sparsey is one such model.¹² In fact, Sparsey started as a purely episodic memory of spatiotemporal patterns and was extended to explain semantic memory (pattern recognition) as well. Sparsey's means for accomplishing classification (semantic memory) is scaffolded on a more primitive and primary episodic memory operation mode.

¹¹ While it may be true that even in the human cognitive system, the details of the individual exemplars of *low-level* object/events, e.g., a particular oriented edge occurring in a particular small patch of the visual field, or a particular edge moving in a certain way in that patch, may be lost (inaccessible), this becomes less true as we consider objects/events of progressively higher scales. This could be taken as evidence for different learning/memory mechanisms operating at low and high scales. However: a) there is a continuum of scales, not just two; and b) there could be one underlying mechanism whose phenomenology changes (gradually) with scale. Sparsey is such a mechanism.

¹² There are other theories addressing both types of memory, but in general, either: a) they have not dealt natively with spatiotemporal patterns; or b) they involve physically distinct subsystems that communicate with each other, e.g., the complementary learning systems (CLS) model of McClelland & O'Reilly.

8. RECOMMENDATIONS

We believe that the following steps would have high likelihood of boosting Sparsey's performance into the SOA range, ~90-100%.

1. Allow RF radius parameters to vary within a level: Adding this variability would make it far easier to find mac grid tilings at any given level, which maximize the fraction of the input feature (pixel) instances, across all frames of all snippets, represented by that level. Ideally, this “effective features” statistic should remain as close as possible to 100% throughout all levels of a network. To the extent this statistic drops with level, the codes at said level, are being chosen without using the information contained in the “dropped features”. There are several ways in which we can maximize effective features, but there are costs associated with them. By allowing neighboring macs to have different size RF radii, we can more easily achieve 100% effective features through all levels without incurring (or at least with minimizing) these costs.
2. Add mac construction mode in which several parameters are sampled from distributions in correlated fashion: Our simulation platform currently has the ability to assign π bound parameters from a specified distribution. A different distribution can be specified for each level of the network. We could add this capability to other parameters fairly easily. These overall list of such parameters could include:
 - π bounds
 - RF radiuses (H, U, and D)
 - Sharpening exponents (H, U, and D)
 - Normalization cutoffs (min and max, for H, U, and D)

In addition to simply adding the ability for these other parameters to be selected from specifiable distributions, we would implement a meta-level protocol to ensure that the set of parameters chosen for any one mac are correlated in appropriate ways. For example, the larger the RF radius, the larger the π bounds should be. Essentially, this protocol would reduce to introducing meta-parameters. This gives the model more flexibility in terms of the space of functions that it can learn.

3. Implement analogs of “ventral” and “dorsal” visual pathways: This is not the same thing as implementing bi-modal input to the model itself. We already have a task, still incomplete, in which we will investigate fusing two visual input sources, an edge input and a HOF input. In this case, the motion features, in the form of HOFs, are computed in preprocessing, i.e., external to the model. While this holds considerable promise to improve performance, it is not our preferred ultimate solution. Ultimately, we want our model to rely only on inputs similar to the inputs that cortex receives, i.e., from thalamus. There are multiple broad output signals from thalamus, minimally, the “parvo” and “magno” channels. In some sense, the “parvo” can be viewed as analogous to our edge input stream and the “magno” can be viewed as analogous to a HOF stream. In any case, cortical V1 receives parvo and magno and they are likely heavily mixed. V1 then gives rise to the dorsal (optimized for form recognition) and ventral (optimized for motion and broad arrangement).

We propose to construct models that explicitly have a V1 which outputs to separate areas analogous to V2 and MT, both of which then project to a single higher area. This is a simplification with respect to actual cortex, but we believe such an architecture may further optimize performance. In our hands, the mac parameters of the “V2” and “MT” analogs would be set in broad analogy to these cortical areas.

4. Investigating a range of data augmentation techniques: We currently use only a very simple form of augmentation: we create five slightly noisy variants of each input and use those to train the model as well.
5. Improve the learning rule: Currently, when an association is made from a code ϕ_A to another code ϕ_B , the ages of the two codes, i.e., how they have been active, are not used to influence the strength of the association. However, doing so would likely improve performance. For example, a level J code ϕ_T^J associates with two subjacent codes ϕ_T^{J-1} and ϕ_{T+1}^{J-1} that become active in sequence. We can imagine that increasing the weights from ϕ_T^J to ϕ_T^{J-1} less than from ϕ_T^J to ϕ_{T+1}^{J-1} would help to correctly disambiguate future similar instances in which time-warping has occurred.
6. Implement user-settable learning parameter specification in configuration files: This would allow us to more easily include these parameters in our large-scale parameter search process.
7. Slaving dynamics at lower levels to familiarity measures at higher levels: We have long wanted to investigate this principle. There is considerable evidence that the brain uses this principle. Specifically, the neuromodulatory systems relevant to detecting novelty vs. familiarity and for setting dynamics appropriately project to all cortical levels, but receive preponderantly from the highest areas. In our model’s terminology, we would say that a G value (or average of G’s) produced in one or macs at a higher level is driving a global sigmoid modulation function which is applied (broadcast) to macs at all levels.
8. Quantify correlation between classification accuracy and trace accuracy: As reported in TR23 and prior reports, we see that classification accuracy can be quite high even when trace accuracy (averaged across all macs at all levels) is quite low, i.e., when there are numerous errors at the single-unit level. In fact, one can see that there are numerous single-unit errors (red cells) in the L2 macs in Figure 1 (due to scaling in the figure, we cannot show the exact codes in the L1 macs, but they too contained numerous unit-level errors). We would like to more systematically quantify the relation of classification accuracy to trace accuracy (and to trace accuracy at each of the individual internal levels). We offered a partial explanation of the reason for this in the prior report, referring to it as the DCCI effect. But there are additional principles/mechanisms underlying the robustness of the class accuracy despite low trace accuracy. We would like to explore them more fully.
9. Explore the parameters of the Backoff Mechanism in detail: During recognition, the model can be set to use “Backoff” or not. Backoff is an automated procedure for choosing which version of the G measures available to a mac to use to choose the code. In general, it seems like the best choice is to use the highest-order G available, where the order is the number of evidence sources being multiplied to produce the distributions (in each CM) from which a winner is chosen. For example, if a mac has active U, H, and D inputs, then it seems plausible that the best estimates of the distributions would result from combining all three

evidence sources: in this case, the V_{HUD} values would be used, resulting in the G_{HUD} version of G .

However, if the input space is such that speed-ups or slow-downs of spatiotemporally localized subsets of overall spatiotemporal inputs, relative to prior instances of such overall inputs, occur appreciably frequently, then it may often occur that lower-order G 's than the highest order G available yield higher values. If a lower order G attains a value significantly higher than the higher order G , this can indicate that the current input instance is equivalent to a prior instance under one or more highly structured transforms, e.g., translation, rotation, omission or insertion of whole parts. In such cases, it may lead to higher recognition accuracies at the scale of the overall inputs than if all macs were always forced to always use the highest-order G available. In other words, backoff is part of a general strategy and set of principles for invariant recognition. That said, while we have mechanism of backoff in place and operating, tuning the many parameters that specify it is still very much an open issue/question. While some general principles suggest the broad form of the backoff policy (i.e., the schedule of backoff thresholds), we have large uncertainty as to the optimal specific parameter values. Moreover, these parameters can vary by level.

8.1 Applying Supervision at Multiple Scales

Stepping back to Section 5.6, we emphasize that while the two instances of digit '4' in Figure 37 may look pretty similar to a human, Sparsey will assign very different L3 codes to them. Without additional supervisory information and a framework for using that information, Sparsey would not capture the fact that these two instances are of the same category. By the same token, the reader has likely readily concurred that the two patterns in M_3^2 's U-RF in Figure 38 (highlighted in orange glow) can be considered instances of the class, "T Junction". However, as can be seen in comparing the violet highlighted codes in Figure 36 and Figure 39, Sparsey assigns very different codes to these two instances. Moreover, a human might easily agree that the two patterns that occur in M_{13}^1 's U-RF (highlighted with rose glow) constitute instances of the class "Horizontal Edge"; the instance in Figure 37a is not perfectly horizontal—the rightmost pixel is out of line with the other three—but at that tiny scale (a 4x4 pixel aperture), this could be easily be construed as simply a noisy version of a perfect horizontal. Yet as can be seen by comparing the codes assigned to these two instances (gray highlights in Figure 36 and Figure 39), the codes assigned to the two "Horizontal Edge" instances are very different.

The point we are building towards here is that supervisory information, i.e., category labels, can potentially be used at multiple conceptual/featural scales. Certainly, supervised learning is applied at progressively larger scales in the case of human learning. Even as infants, our behavior is channeled (supervised) in numerous ways. Generally, teaching signals given to infants, concern relatively small-scale features. In the early years, we are explicitly taught how to draw straight line segments, curves, etc. When we are a little older we begin to be taught letters, including in terms of the basic strokes of which they are composed, then pairs/groups of letters, then words, etc.

We did not follow this rubric in the experiments of Section 5.6: rather, we applied supervised learning only at the L3 scale, as shown in Figure 35. However, in future research, we plan to experiment with applying supervision at a progression of scales, as suggested in Figure 66. First, we would teach the network, or rather, individual L1 macs, the concepts of ‘vertical edge’, ‘horizontal edge’, ‘diagonal’, etc. Once the L1 macs have learned their features sufficiently well, we would begin providing labels to L2 macs, then at the L2 level, and then at L3. However, the model also allows supervised learning to be applied in interleaved fashion across levels. There is no strict requirement for proceeding upward, level by level. However, note that even if we apply supervised learning at L1, unsupervised learning will still occur throughout all levels of the network. That is, codes in macs at L2 and higher will be learned (stored) from the very first learning trials presented to the model.

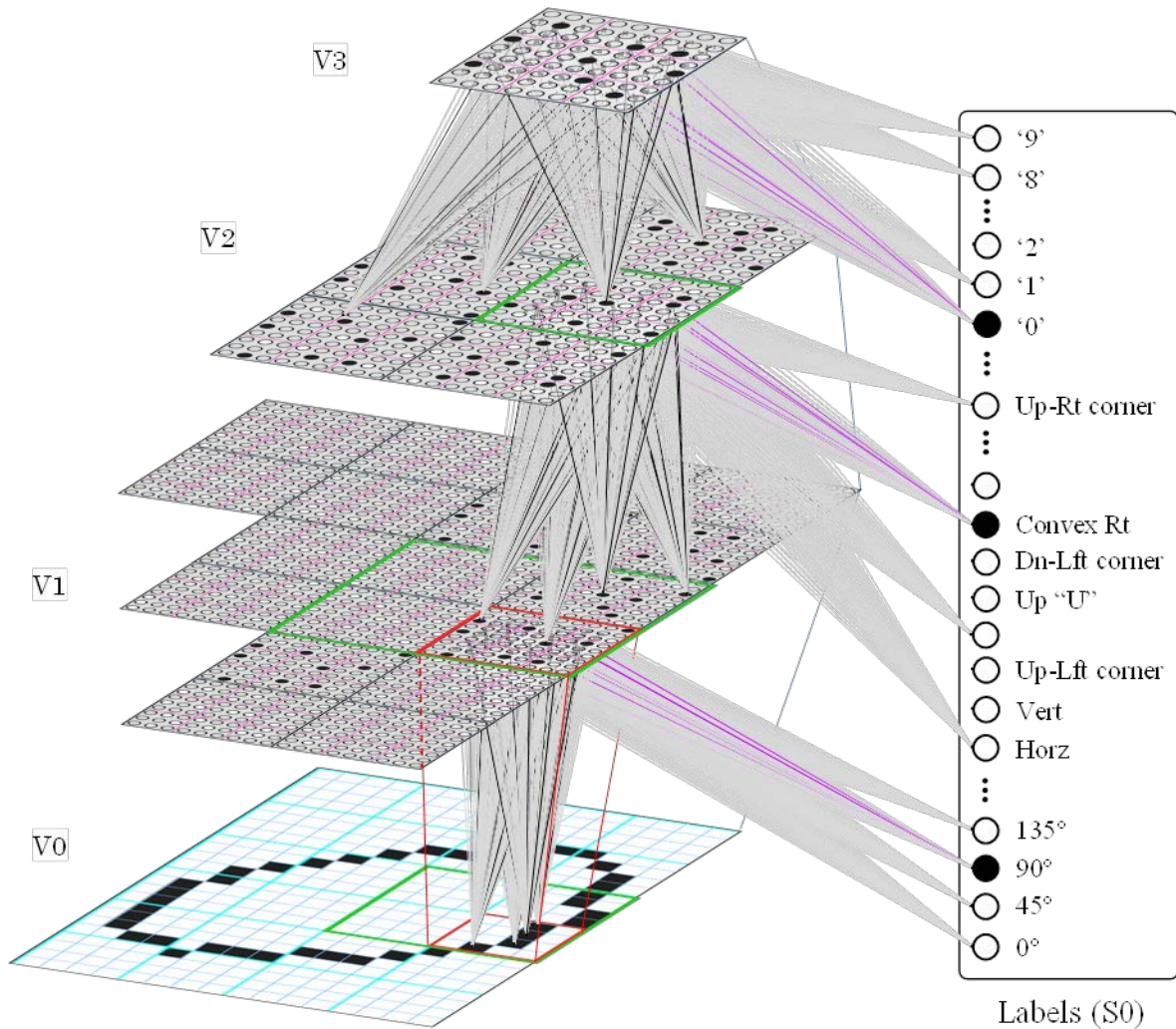


Figure 66: Illustration of Applying Supervised learning at Multiple Scales in the Network of Figure 35

Thus, we emphasize that Figure 66 simply suggests some aspects of a possible overall architecture. The operational details of multiple-scale, interleaved supervised learning need to be developed. A key question is: how would the results of supervised learning at a lower scale, e.g., L1 (V1), be used to abet/guide learning at higher levels, e.g., L2, L3, etc.? And, this has several sub-questions: how would it abet/guide:

1. The *unsupervised* learning of higher level SDC codes in the macs of a sensory (non-symbolic, non-label-representing) network?
2. The *supervised* learning of such codes, i.e., the association of higher-level mac codes to representations of other categories?
3. The unsupervised learning of codes in macs at any hierarchical level in other sensory networks of other modalities, e.g., auditory, somatic, etc.

Regarding the first question, we described in Sec. VI of TR4 two different supervised learning modes, Sup-mode 1 and Sup-mode 2. Thus far, our studies have only involved Sup-mode 1, in which the supervised learning is only from the L3 (V3) codes to the category codes. Thus, there is no influence of previously acquired category information on the choice of subsequent SDC codes. In other words, U-signals from the category codes to the sensory codes are not used in choosing winners in the CMs of macs. However, they could be, and in natural systems probably are, an important influence on the learning process. We intend to implement and explore this principle, i.e., Sub-mode 2, in the near future.

8.2 Outstanding Questions Regarding Parameter Settings

8.2.1. Backoff

During recognition, the model can be set to use “Backoff” or not. Backoff is an automated procedure for choosing which version of the G measures available to a mac to use to choose the code. In general, it seems like the best choice is to use the highest-order G available, where the order is the number of evidence sources being multiplied to produce the distributions (in each CM) from which a winner is chosen. For example, if a mac has active U, H, and D inputs, plausibly the best estimates of the distributions would result from combining all three evidence sources: in this case, the V_{HUD} values would be used, resulting in the G_{HUD} version of G .

However, suppose that the input space is such that speed-ups or slow-downs of spatiotemporally localized subsets of overall spatiotemporal inputs, relative to prior instances of such overall inputs, occur appreciably frequently. In that case, it may often occur that lower-order G ’s than the highest order available yield higher G (familiarity) values than the higher order G ’s. In such cases, if a lower order G attains a value significantly higher than the higher order G , this can indicate that the current input instance is equivalent to a prior instance under one or more highly structured transforms, e.g., translation, rotation, omission or insertion of whole parts. In such cases, it may lead to higher recognition accuracies at the scale of the overall inputs than if all macs were always forced to use the highest-order G available. In other words, backoff is part of a general strategy and set of principles for invariant recognition. That said, while we have mechanism of backoff in place and operating, tuning the many parameters that specify it is still very much an open issue/question.

At more mundane level, we do not yet have general principles for setting the specific parameters of the backoff procedure. A particular example of a “backoff schedule” is shown below. This says that for macs at level 2, if the highest-order G available is G_{HUD} , then if $G_{HUD} \geq \Gamma_{HUD} = 0.5$, then use G_{HUD} (and the underlying V_{HUD} values to choose the code. Otherwise evaluate the next lower order versions of V and G , i.e., the HU and UD versions, and see if the max of those G’s attains another threshold, $\Gamma_{2-way} = 0.85$. If it does, then use it. Otherwise backoff to the next lower order G, which here would be G_U and see if it passes yet another threshold, $\Gamma_U = 0.96$. While some general principles suggest the broad form of the backoff, we have large uncertainty as to what the optimal specific numeric parameters should be. Moreover, these parameters can vary by level.

9. REFERENCES

- Andoni, A. and P. Indyk (2008). "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions." Communications of the ACM **51**(1): 117-122.
- Barkat, T. R., D. B. Polley and T. K. Hensch (2011). "A critical period for auditory thalamocortical activity". Nature Neuroscience. " Nature Neurosci. **14**(9): 1189-1196.
- Barrett, A. B., G. O. Billings, R. G. M. Morris and M. C. W. van Rossum (2009). "State Based Model of Long-Term Potentiation and Synaptic Tagging and Capture." PLoS Computational Biology **5**(1): e1000259.
- Behrend, D. A., J. Scofield and E. E. Kleinknecht (2001). "Beyond fast mapping: Young children's extensions of novel words and novel facts." Developmental Psychology **37**(5): 698.
- Bengio, Y. (2007). On the challenge of learning complex functions. Progress in Brain Research. P. Cisek, T. Drew and J. F. Kalaska, Elsevier. **Volume 165**: 521.
- Bengio, Y., A. Courville and P. Vincent (2012). Representation Learning: A Review and New Perspectives, U. Montreal.
- Bengio, Y. and Y. leCun (2007). Scaling Learning Algorithms towards AI. Large-Scale Kernel Machines. L. Bottou, O. Chapelle, D. DeCoste and J. Weston, MIT Press.
- Brooks, L. R. (1987). Decentralized Control of Categorization: the role of prior processing episodes. Concepts and Conceptual Development: Ecological and Intellectual Factors in Categorization. Emory Symposia in Cognition 1 (Oct. 1984). U. Neisser, Cambridge University Press: 141-174.
- Carey, S. and E. Bartlett (1978). "Acquiring a single new word." Papers and Reports on Child Language Development **15**: 17-29.
- DeAngelis, G. C., G. M. Ghose, I. Ohzawa and R. D. Freeman (1999). "Functional micro-organization of primary visual cortex: receptive field analysis of nearby neurons." J Neurosci **19**(10): 4046-4064.
- DeAngelis, G. C., I. Ohzawa and R. D. Freeman (1993). "Spatiotemporal organization of simple-cell receptive fields in the cat's striate cortex. I. General characteristics and postnatal development." J Neurophysiol **69**(4): 1091-1117.
- Desimone, R., T. D. Albright, C. G. Gross and C. Bruce (1984). "Stimulus-selective properties of inferior temporal neurons in the macaque." J. Neurosci. **4**(8): 2051-2062.
- DiCarlo, James J., D. Zoccolan and Nicole C. Rust (2012). "How Does the Brain Solve Visual Object Recognition?" Neuron **73**(3): 415-434.
- Dollaghan, C. (1985). "Child meets word: "fast mapping" in preschool children. ." Journal of speech and hearing research **28**(3): 449-454.
- Douglas, R. J., K. A. Martin and D. Witteridge (1989). "A canonical microcircuit for neocortex." Neural Computation **1**(4): 480-488.
- Douglas, R. J. and K. A. C. Martin (2004). "Neuronal Circuits of the Neocortex." Annual Review of Neuroscience **27**(1): 419-451.
- Feldman, V. and L. G. Valiant (2009). "Experience-Induced Neural Circuits That Achieve High Capacity." Neural Computation **21**(10): 2715-2754.
- Frey, B. J. (1998). Graphical models for machine learning and digital communication, MIT Press.
- Frey, U. and R. G. M. Morris (1997). "Synaptic tagging and long-term potentiation." Nature **385**(6616): 533-536.
- Grigorescu, C., N. Petkov and M. A. Westenberg (2004). "Contour and boundary detection improved by surround suppression of texture edges." Image and Vision Computing **22**(8): 609-622.

- Grossberg, S. (1980). "How does a brain build a cognitive code?" Psychological Review **87**(1): 1-51.
- Hebb, D. O. (1949). The organization of behavior. New York, Wiley.
- Hecht-Nielsen, R. (2005). "Cogent confabulation." Neural Networks **18**(2): 111-115.
- Hinton, G. E., S. Osindero and Y.-W. Teh (2006). "A Fast Learning Algorithm for Deep Belief Nets." Neural Computation **18**(7): 1527-1554.
- Hintzman, D. L. (1986). "'Schema abstraction" in multiple-trace memory model " Psychological Review **93**: 411-428.
- Jafarpour, A., L. Fuentemilla, A. J. Horner, W. Penny and E. Duzel (2014). "Replay of Very Early Encoding Representations during Recollection." The Journal of Neuroscience **34**(1): 242-248.
- Ji, D. and M. A. Wilson (2007). "Coordinated memory replay in the visual cortex and hippocampus during sleep." Nat Neurosci **10**(1): 100.
- Jitsev, E. (2010). On the self-organization of a hierarchical memory for compositional object representation in the visual cortex, Goethe University Frankfurt am Main.
- Jockel, S. (2009). Crossmodal Learning and Prediction of Autobiographical Episodic Experiences using a Sparse Distributed Memory. PhD, University of Hamburg.
- Jusczyk, P. W. (1999). "How infants begin to extract words from speech." Trends in Cognitive Sciences **3**(9): 323-328.
- Kanerva, P. (1988). Sparse distributed memory. Cambridge, MA, MIT Press.
- Kanerva, P. (1994). The Spatter Code for encoding concepts at many levels. Proceedings of International Conference on Artificial Neural Networks, Sorento, Italy, Springer-Verlag.
- Kanerva, P. (2009). "Hyperdimensional Computing: An Introduction to Computing in Distributed Representation with High-Dimensional Random Vectors." Cognitive Computing **1**: 139-159.
- Kiani, R., H. Esteky, K. Mirpour and K. Tanaka (2007). "Object Category Structure in Response Patterns of Neuronal Population in Monkey Inferior Temporal Cortex." J Neurophysiol **97**(6): 4296-4309.
- Kouh, M. and T. Poggio (2008). "A Canonical Neural Circuit for Cortical Nonlinear Operations." Neural Computation **20**(6): 1427-1451.
- Kreiman, G., C. Hung, A. Kraskov, R. Q. Quiroga, T. Poggio and J. Dicarlo (2006). "Object selectivity of local field potentials and spikes in the macaque inferior temporal cortex." Neuron **49**: 433-445.
- Kruschke, J. K. (1992). "ALCOVE - An Exemplar-Based Connectionist Model of Category Learning." Psychological Review **99**(1): 22-44.
- Laptev, I., M. Marszalek, C. Schmid and B. Rosenfeld (2008). Learning realistic human actions from movies. CVPR.
- Le, Q. V., W. Zou, S. Yeung and A. Y. I. Ng, 2011 (2011). Learning hierarchical spatio-temporal features for action recognition with independent subspace analysis. CVPR-11.
- LeCun, Y. and Y. Bengio (1995). Convolutional networks for images, speech, and timeseries.
- Lee, H., R. Grosse, R. Ranganath and A. Ng (2011). "Unsupervised Learning of Hierarchical Representations with Convolutional Deep Belief Networks." Communications of the ACM **54**(10): 95-103.
- Litvak, S. and S. Ullman (2009). "Cortical Circuitry Implementing Graphical Models." Neural Computation **21**(11): 3010-3056.
- Liu, J., J. Luo and S. M. (2009). Recognizing realistic actions from videos "in the Wild". CVPR.

Luczak, A., B. L. McNaughton and K. D. Harris (2015). "Packet-based communication in the cortex." Nat Rev Neurosci **16**(12): 745-755.

Marszalek, M., I. Laptev and C. Schmid (2009). Actions in Context. CVPR.

McKenzie, S., Andrea J. Frank, Nathaniel R. Kinsky, B. Porter, Pamela D. Rivière and H. Eichenbaum (2014). "Hippocampal Representation of Related and Opposing Memories Develop within Distinct, Hierarchically Organized Neural Schemas." Neuron **83**(1): 202-215.

Moll, M. and R. Miiikkulainen (1995). Convergence-Zone Episodic Memory: Analysis and Simulations, University of Texas at Austin, Dept. of Computer Science.

Moll, M. and R. Miiikkulainen (1997). "Convergence-Zone Episodic Memory: Analysis and Simulations." Neural Networks **10**(6): 1017-1036.

Moncada, D. and H. Viola (2007). "Induction of Long-Term Memory by Exposure to Novelty Requires Protein Synthesis: Evidence for a Behavioral Tagging." J. Neurosci. **27**(28): 7476-7481.

Morris, R. G. M. and U. Frey (1999). "Tagging the Hebb synapse: Reply." Trends in Neurosciences **22**(6): 256.

Murray, J. F. and K. Kreutz-Delgado (2007). "Visual Recognition and Inference Using Dynamic Overcomplete Sparse Learning." Neural Computation **19**(9): 2301-2352.

Nandy, Anirvan S., Tatyana O. Sharpee, John H. Reynolds and Jude F. Mitchell (2013). "The Fine Structure of Shape Tuning in Area V4." Neuron **78**(6): 1102-1115.

O'Neill, J. and J. Csicsvari (2014). "Learning by Example in the Hippocampus." Neuron **83**(1): 8-10.

Pandipati, S. and N. E. Schoppa (2012). "Age-dependent adrenergic actions in the main olfactory bulb that could underlie an olfactory-sensitive period." Journal of Neurophysiology **108**(7): 1999-2007.

Pitkow, X. (2012). Compressive neural representations of sparse, high-dimensional probabilities. Advances in Neural Information Processing Systems.

Pouget, A., J. M. Beck, W. J. Ma and P. E. Latham (2013). "Probabilistic brains: knowns and unknowns." Nat Neurosci **16**(9): 1170-1178.

Rachkovskij, D. A. (2001). "Representation and Processing of Structures with Binary Sparse Distributed Codes." IEEE Transactions on Knowledge and Data Engineering **13**(2): 261-276.

Rachkovskij, D. A. and E. M. Kussul (2001). "Binding and Normalization of Binary Sparse Distributed Representations by Context-Dependent Thinning." Neural Computation **13**: 411-452.

Rachkovskij, D. A. and E. M. Kussul (2001). "Binding and Normalization of Binary Sparse Distributed Representations by Context-Dependent Thinning." Neural Computation **13**(2): 411-452.

Ramirez, A., E. A. Pnevmatikakis, J. Merel, L. Paninski, K. D. Miller and R. M. Bruno (2014). "Spatiotemporal receptive fields of barrel cortex revealed by reverse correlation of synaptic input." Nat Neurosci **17**(6): 866-875.

Riesenhuber, M. and T. Poggio (1999). "Hierarchical models of object recognition in cortex." Nat Neurosci **2**(11): 1019.

Rinkus, G. (1996). A Combinatorial Neural Network Exhibiting Episodic and Semantic Memory Properties for Spatio-Temporal Patterns. Ph.D., Boston University.

Rinkus, G. J. (2010). "A cortical sparse distributed coding model linking mini- and macrocolumn-scale functionality." Frontiers in Neuroanatomy **4**.

Rodriguez, M., J. Ahmed and M. Shah (2008). Action mach: A spatio-temporal maximum average correlation height filter for action recognition. ICCV.

Rust, N. C. and J. J. DiCarlo (2010). "Selectivity and Tolerance ("Invariance") Both Increase as Visual Information Propagates from Cortical Area V4 to IT." The Journal of Neuroscience **30**(39): 12978-12995.

Sajikumar, S. and J. U. Frey (2004). "Resetting of `synaptic tags' is time- and activity-dependent in rat hippocampal CA1 in vitro." Neuroscience **129**(2): 503-507.

Salakhutdinov, R. and G. Hinton (2009). "Semantic hashing." International Journal of Approximate Reasoning **50**(7): 969-978.

Saul, L. K. and S. Roweis (2002). Think Globally, Fit Locally: Unsupervised Learning of Nonlinear Manifolds, U. Penn.

Schuldt, C., I. Laptev and B. Caputo (2004). Recognizing Human Actions: A Local SVM Approach. ICPR, Cambridge, UK.

Serre, T., M. Kouh, C. Cadieu, U. Knoblich, G. Kreiman and T. Poggio (2005). A Theory of Object Recognition: Computations and Circuits in the Feedforward Path of the Ventral Stream in Primate Visual Cortex. AI Memo 2005-036, MIT.

Serre, T., G. Kreiman, M. Kouh, C. Cadieu, U. Knoblich, T. Poggio and T. D. a. J. F. K. Paul Cisek (2007). A quantitative theory of immediate visual recognition. Progress in Brain Research, Elsevier. **Volume 165**: 33.

Smith, L. and C. Yu (2008). "Infants rapidly learn word-referent mappings via cross-situational statistics." Cognition **106**(3): 1558.

Taylor, G. W., R. Fergus, Y. LeCun and C. Bregler (2010). Convolutional Learning of Spatio-temporal Features. European Conference on Computer Vision (ECCV'10).

Theunissen, F. E. and J. E. Elie (2014). "Neural processing of natural sounds." Nat Rev Neurosci **15**(6): 355-366.

Tulving, E. (1972). Episodic and Semantic Memory. Organization of Memory. E. Tulving and W. Donaldson. New York, Academic Press.

Tulving, E. (1983). Elements of Episodic Memory. Oxford Oxford University Press.

Valiant, L. (2006). "A quantitative theory of neural computation." Biological Cybernetics **95**(3): 205-211.

Vokey, J. R. and L. R. Brooks (1994). "FRAGMENTARY KNOWLEDGE AND THE PROCESSING-SPECIFIC CONTROL OF STRUCTURAL SENSITIVITY." Journal of Experimental Psychology-Learning Memory and Cognition **20**(6): 1504-1510.

Whittlesea, B. W. A. and M. D. Dorken (1993). "Incidentally, things in general are particularly determined - An episodic-processing account of implicit learning." Journal of Experimental Psychology-General **122**(2): 227-248.

Wiesel, T. N. and D. H. Hubel (1963). "Effects of visual deprivation on morphology and physiology of cell in the cat's lateral geniculate body." J. Neurophys. **26**(6).

Willshaw, D. J., O. P. Buneman and H. C. Longuet-Higgins (1969). "Non Holographic Associative Memory." Nature **222**: 960-962.

Wilson, M. A. and B. L. McNaughton (1994). "Reactivation of hippocampal ensemble memories during sleep." Science **265**(5172): 676-679.

Wittgenstein, L. (1953). Philosophical Investigations, Blackwell.

Zeiler, M. D., S. F. Taylor and R. Fergus (2011). Adaptive Deconvolutional Networks for Mid and High Level Feature Learning. ICCV-13.





APPENDIX A - Edge Filtering Detail

The video datasets contain sequences of humans performing natural actions and movements characteristic to many sports and activities. In order to accomplish the project's goals, we preprocess all the videos within each dataset to standardize the spatial resolution, perform edge detection, and binarize. This preprocessing results in visual input features similar to the output of the lateral geniculate nucleus (LGN) in the primate thalamus. First, we describe the video datasets. Second, we report the video processing protocol that we developed to detect edges and binarize videos in the datasets. Sample frames are shown throughout the presentation of the video processing methodology with different parameter settings.

A.1 The Video Datasets

We consider the KTH dataset (Schuldt, Laptev et al. 2004), UCF sports action dataset (Rodriguez, Ahmed et al. 2008), YouTube action dataset (Liu, Luo et al. 2009), and the Hollywood 2 dataset (Marszalek, Laptev et al. 2009). Table 17 depicts representative sample frames from each dataset and describes characteristic actions contained within each. The KTH dataset consists of video sequences of 25 human subjects in 4 indoor and outdoor scenes, performing 6 actions. The UCF dataset consists of 200 video sequences of human subjects in outdoor scenes performing 9 categories of actions. The YouTube action dataset consists of 1100 video sequences: 11 human action categories such as diving, cycling, golf swinging, and horseback riding, 25 action type groupings, and 4 clips in each group. Finally, the Hollywood 2 dataset is composed of 3669 action clips of human actions and scenes from Hollywood films. There are 12 classes of human actions, such as kissing, picking up a phone, and sitting, and 10 classes of scenes, such as bedroom, hotel, kitchen, and office. Videos in each dataset vary in encoding scheme, color depth, and spatial resolution. Before edge detection and binarization, we standardized all videos by converting them to 8 bit 300x240 grayscale sequences in avi format.


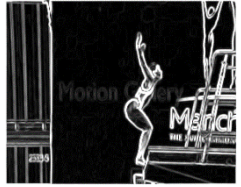
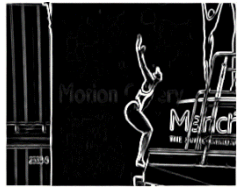



Table 17. Video Datasets

Video Dataset	Actions	Sample Frames
KTH (Schuldt, Laptev et al. 2004)	Running, boxing, hand waving, walking, etc.	
UCF Sports Action (Rodriguez, Ahmed et al. 2008)	Golf swings, biking, diving, horse riding, etc.	
Youtube Action (Liu, Luo et al. 2009)	Shooting, cycling, juggling, kicking, etc.	
Hollywood 2 (Marszalek, Laptev et al. 2009)	Answering phone, hand shaking, sitting, eating, kissing, etc.	

A.2 Preprocessing Protocol

We implemented an automated routine in MATLAB to recursively traverse a folder hierarchy of each dataset, locate all video files, convert the videos into 8 bit 300x240 grayscale videos, and detect edges in each video. The outputs are uncompressed binarized videos that TEMECOR can accept as input. Table 18 presents an overview of video processing cascade. We now present each processing stage in detail.

Table 18. Overview of Video Processing Stages to Detect Edges and Binarize Videos

Stage	Technique	Description	Sample Frame
1	Gaussian filtering	Spatial smoothing for noise reduction	
2	Gabor edge detection	Extract 112 grayscale maps (16 edge orientations, 7 spatial phases)	
3	Orientation and phase superposition	Obtain grayscale gradient image by taking the L^2 -norm of orientation and L^∞ -norm phase maps.	
4	Surround Suppression	Suppress regions in each frame that contain high spatial frequency textures are not central to the action being performed in the scene. The technique reduces noise and variance across consecutive frames.	
5	Edge thinning (non-maxima suppression)	Reduce the width of image contours to one pixel.	
6	Hysteresis thresholding	Binarize video frames based on neighborhood connectivity and contour magnitude	
7	Suppression Slope	Suppress surviving contours based on the likelihood they are texture edges, and not central to the action taking place in the video.	

1. Stage 1: Gaussian Smoothing

In the first stage, we perform Gaussian filtering of each frame of video. We found that this smoothing procedure improved edge detection performance. The next stage computes gradients in each image frame, which is prone to video compression artifacts, discretization and noise, since differences are computed over small neighborhoods of the frame. Low pass filtering using a Gaussian filter reduces the subsequent effects of these artifacts. We used the following normalized 2D Gaussian filter G in computations:

$$G_u(x, y) = \frac{e^{-\frac{x^2+y^2}{2\sigma^2}}}{\sqrt{2\pi}\sigma}$$

A-1

$$G(x, y) = \frac{G_u(x, y)}{\sum_x \sum_y G_u(x, y)}$$

Our Gaussian filters had a 5x5 pixel size and we fixed $\sigma = 1.5$, and x and y correspond to spatial positions of pixels within the filter neighborhood.

2. Stage 2: Gabor Edge Detection

In this stage, we compute image gradients by filtering each frame with Gabor wavelets that have 7 spatial phases and 16 orientations. Figure 67 depicts a subset of the filters with 6 different orientations (top row) and 6 different phases (bottom row).

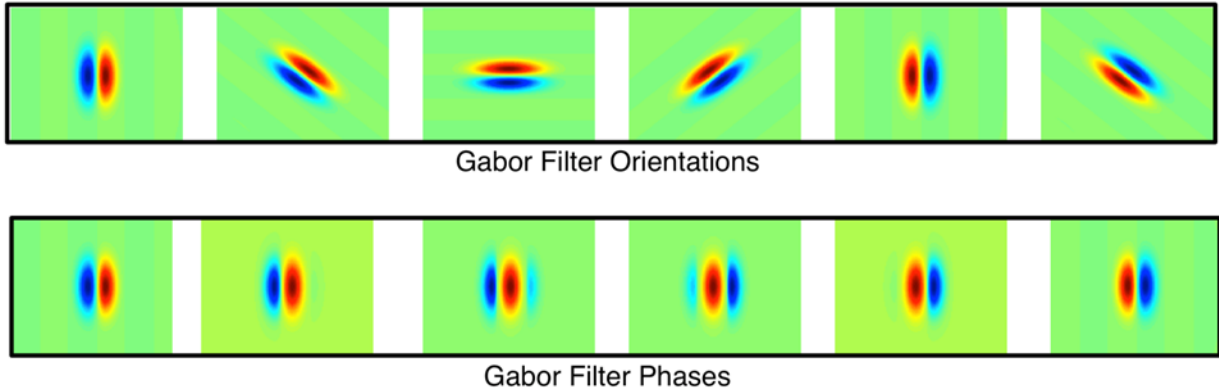


Figure 67: Example Gabor Wavelets Used in Our Preprocessing

Gabor filters are defined according to Eq. A-2:

$$B(x, y) = \cos\left(\frac{2\pi x'}{\lambda} + \varphi\right) e^{-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}} \quad \text{A-2}$$

$$x' = x \cos(\theta) + y \sin(\theta)$$

$$y' = -x \sin(\theta) + y \cos(\theta)$$

In Eq. A-2, λ defines the filter wavelength in pixels, φ is the spatial phase, σ signifies the standard deviation of the Gaussian component, γ corresponds to the filter aspect ratio, (x', y') is the rotation matrix that aligns the Gabor filter in the direction of the orientation θ . The value for parameter σ is specified directly according to the spatial bandwidth b , defined by Eq. A-3, which describes the spatial frequency of the half-maximum response of the Gabor filter.

$$\sigma = \frac{\lambda}{\pi} \sqrt{\frac{\text{Log}(2)}{2} \frac{2^b + 1}{2^b - 1}} \quad \text{A-3}$$

The Gabor filter size is automatically determined according to the ratio $\frac{\sigma}{\gamma}$. We set default parameters to $\lambda = 3$, $\gamma = 0.5$, $b = 10$. The result of convolution of each video frame with the Gabor filter bank \mathbf{B} is 112 feature maps (16 orientations x 7 phases). The next stage specifies how the extracted features are combined to form a single image gradient map of each frame.

3. Stage 3: Orientation and Phase Superposition

Orientation and phase superposition refers to the combination of the corresponding feature maps into a single image gradient. Orientation features are merged according to the L^2 -norm, i.e. $\sqrt{X^2 + Y^2 + \dots}$ for orientation feature maps X, Y, \dots . According to this orientation superposition scheme, edges detected at a number of orientations factor into the final value at each pixel. We use the L^∞ -norm to combine phases, i.e. $\max(X, Y, \dots)$, under the assumption that any particular edge in the video frame will have a dominant spatial phase. Figure 68 shows the detected gradient maps on frames at the beginning, middle, and end of sample videos. Figure 69 depicts how different values of λ , γ , and b affect the gradient map of a sample frame of video.

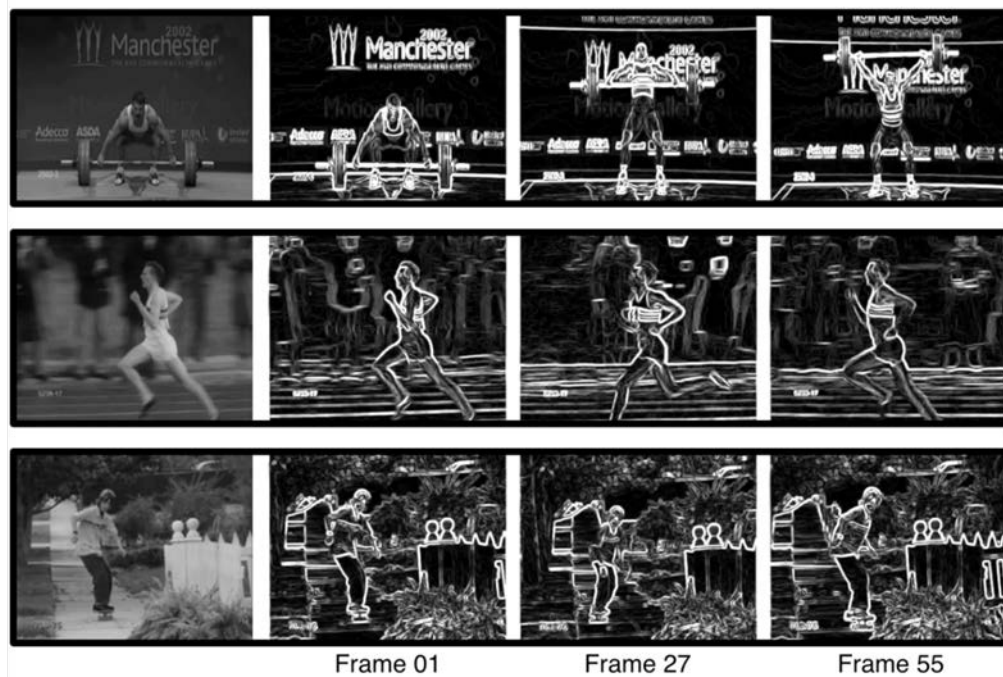


Figure 68: Gradient Image Frames Computed by Gabor Wavelet Filtering

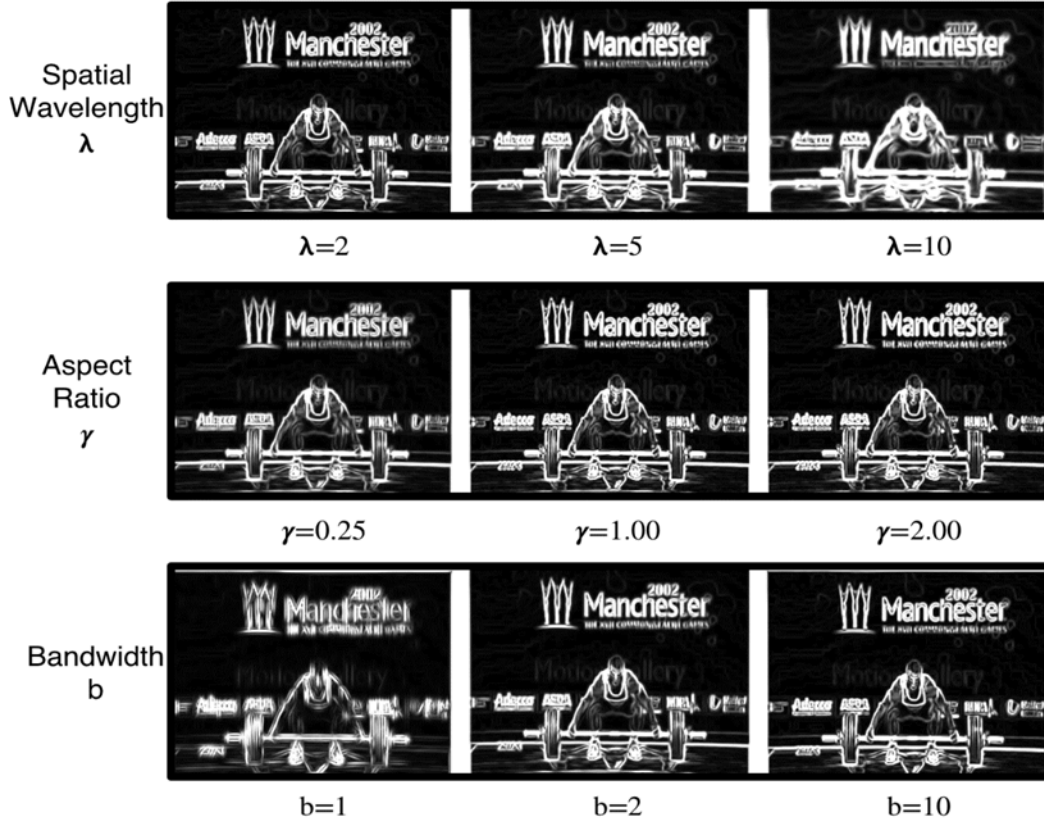


Figure 69: Gabor Wavelet Image Gradients with Different Parameter Values for the Spatial Wavelength, Aspect Ratio, and Bandwidth

4. Stage 4: Surround Suppression

In this stage, we implement a technique to improve the quality of edges that are detected in video frames called *surround suppression* (Grigorescu, Petkov et al. 2004). The operation enhances the response to isolated region boundaries, edges due to objects, and lines, but weakens the response to video artifacts, noise, and jagged texture edges. The approach is motivated by the finding in primate primary visual cortex (V1) that neurons demonstrate suppression if a stimulus appears outside the classical receptive field (Figure 70). Texture in the surrounding neighborhood of pixel (x,y) (orange) will be suppressed. First, an annulus-shaped kernel is constructed to select the values surrounding each pixel in the gradient map. We implement the annular kernel through a difference-of-Gaussians (DoG) (Eq. A-4) followed by a normalization and half-wave rectification stage (Eq. A-5).

$$DoG_{\sigma_d}(x, y) = \frac{1}{2\pi(4\sigma_d)^2} e^{-\frac{x^2+y^2}{2(4\sigma_d)^2}} - \frac{1}{2\pi(\sigma_d)^2} e^{-\frac{x^2+y^2}{2(\sigma_d)^2}} \quad A-4$$

$$w_{\sigma_d}(x, y) = \frac{\max(DoG_{\sigma_d}(x, y), 0)}{\|\max(DoG_{\sigma_d}(x, y), 0)\|_1} \quad A-5$$

In Eq. A-5, we fix $\sigma_d = 1.6$, and $\|X\|_1$ refers to the L^1 -norm.

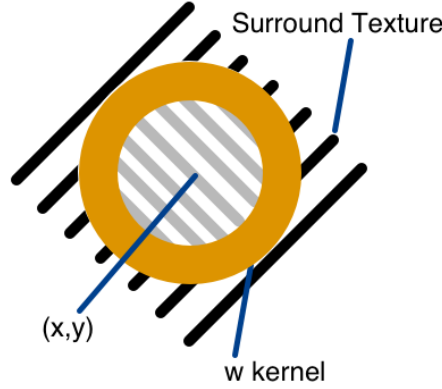


Figure 70: Illustration of Texture Suppression Kernel w_σ

Next, we convolve the image gradient M of each video frame with the on-surround kernel w_{σ_d} in Eq. A-6 and obtained the suppressed grayscale gradient map S_{σ_d} by applying the suppression strength factor α , which we fix to 0.6.

$$t_{\sigma_d}(x, y) = \iint M_{\sigma_d}(x - u, y - v) w_{\sigma_d}(u, v) du dv$$

$$S_{\sigma_d} = \max(M_{\sigma_d}(x, y) - \alpha t_{\sigma_d}(x, y), 0) \quad \text{A-6}$$

Figure 71 shows the performance of surround suppression on the sample video clip with different settings for suppression strengths (α). Edges non-essential to the action taking place in the scene are suppressed, while the edges of the human performing the action are preserved.

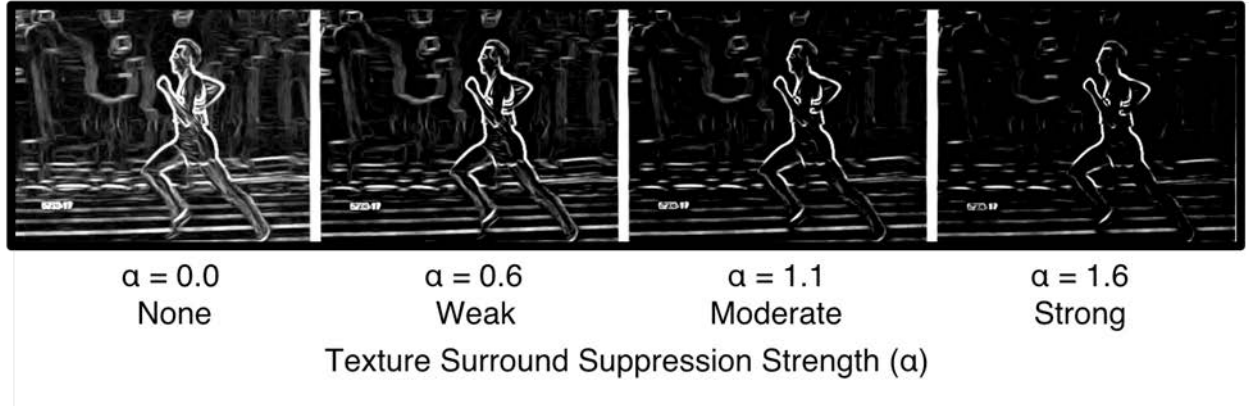


Figure 71: Effects of Texture Surround Suppression on Image gradient of Sample Frame

5. Stage 5: Edge Thinning

Stage 5 thins contours present in the output of the surround suppression stage (S_{σ_d}) to be one pixel wide. The approach is called *non-maxima suppression*. From the orientation maps computed during Stage 2, we construct a map for each video frame that labels the orientation that garnered the largest value at each pixel location. This tells us the local contour orientation if there a contour of an object in the scene that passes through location (x,y). From the contour orientation estimates at each location, we interpolate a line in the specified direction in the local pixel neighborhood. If the central pixel magnitude is greater those of the surrounding pixels in the neighborhood that intersect the line, then the central pixel value is preserved. However, if the neighboring pixels on the line are greater, the central pixel is suppressed. This thinning is efficiently implemented by shifting the entire frame in the direction of the dominant local pixel orientation and comparing the magnitude central and neighboring pixel values. Figure 72 displays frames from video clips following non-maxima suppression edge thinning.

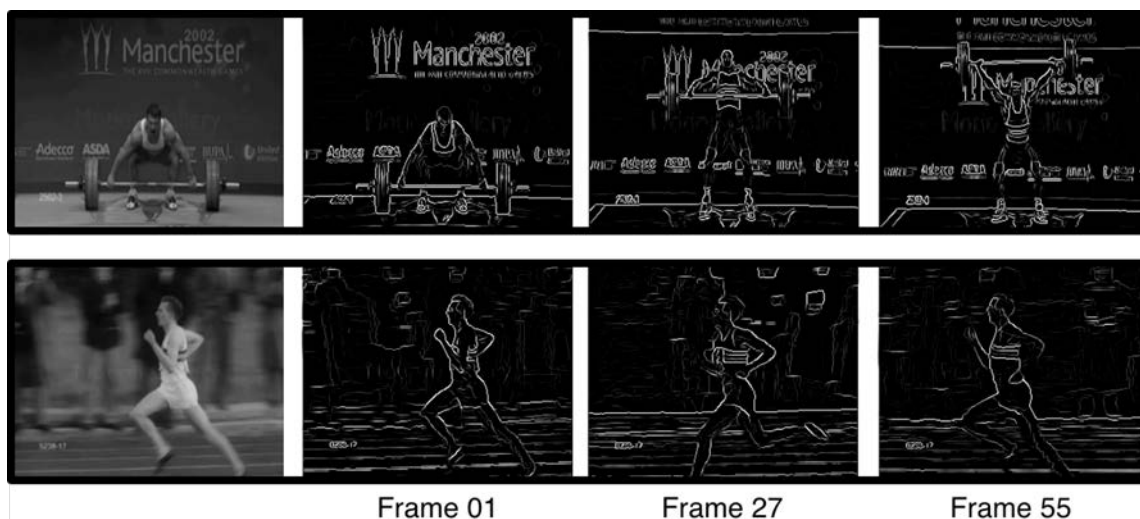


Figure 72: The Results of Edge Thinning by Non-Maxima Suppression

6. Stage 6: Hysteresis Thresholding

The output of Stage 6 is a binary image of each video frame, which is achieved by a hysteresis thresholding procedure based on the strengths of contours computed in Stage 5. There is a low threshold t_{l_1} below which pixels in the output of Stage 5 are set to zero, and a high threshold t_{h_1} above which pixels are preserved. Intermediate pixel values between the two thresholds are preserved when they border candidate pixels with values greater than t_{h_1} , otherwise they are set to zero. In other words, candidate binary edge pixels need to have active neighbors that connect them to a larger edge in the image. We compute the thresholds according to the image statistics: t_{l_1} is set to the 0.25-quantile of pixel values, and t_{h_1} is set to $2t_{l_1}$. This approach confers the advantage that the thresholds generalize to videos with very different content and adaptively threshold video frames based on the distribution of pixel values. Figure 73 shows sample binarized video frames.

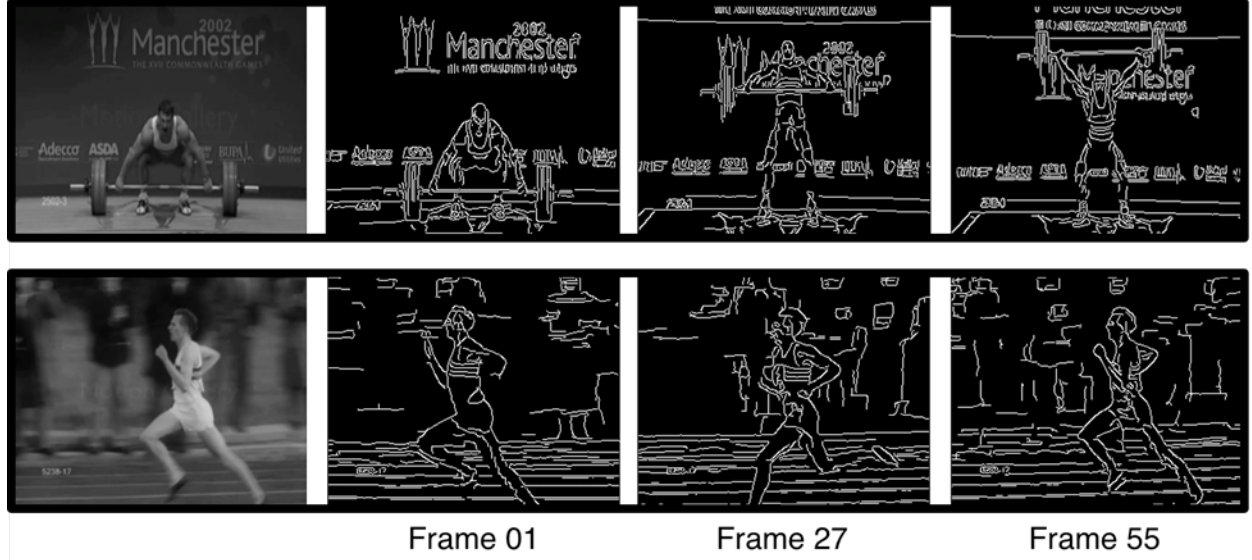


Figure 73: Binarization Results Following Hysteresis Thresholding

7. Stage 7: Suppression Slope

This stage applies a post-processing enhancement to the binary videos called the suppression slope (Grigorescu, Petkov et al. 2004). In Stage 4, we performed surround suppression to mitigate the appearance of texture in the image gradients. From Eq. A-6, t_{σ_d} correlates with how much texture exists around each pixel: higher values indicate a higher likelihood that the pixel was surrounded by texture, while lower values indicate a low chance the pixel is surrounded by a texture. By computing the derivative, the suppression slope, we can quantify the probability that each pixel is a texture pixel and not part of a more global smooth contour. Eq. A-7 defines the suppression slope E as proportional to the ratio between the surround suppression strength t_{σ_d} and the image gradient M_{σ_d} :

$$E(x, y) = \arctan \left(\frac{t_{\sigma_d}(x, y)}{M_{\sigma_d}(x, y)} \right) \quad \text{A-7}$$

To implement a strategy that preserves pixel values that have low suppression slopes (i.e., are not texture elements) and set pixel values that have high suppression slopes to zero (i.e., are likely texture elements), we apply a second hysteresis thresholding stage based on the values of E computed across each video frame. There is a low threshold t_{l_2} below which pixels in the output of Stage 5 are preserved, and a high threshold t_{h_2} above which pixels are set to zero. The low threshold t_{l_2} is set to the 0.25-quantile value among the distribution of suppression slope values, and $t_{h_2} = 2t_{l_2}$.

The final output of the video processing cascade is given by the intersection of binary outputs computed in Stages 6 and 7 (i.e., their product). Sample frames of the final binary images are presented in Figure 74. Figure 75 compares the outputs of Stages 6 and 7, without and with the suppression slope, respectively.

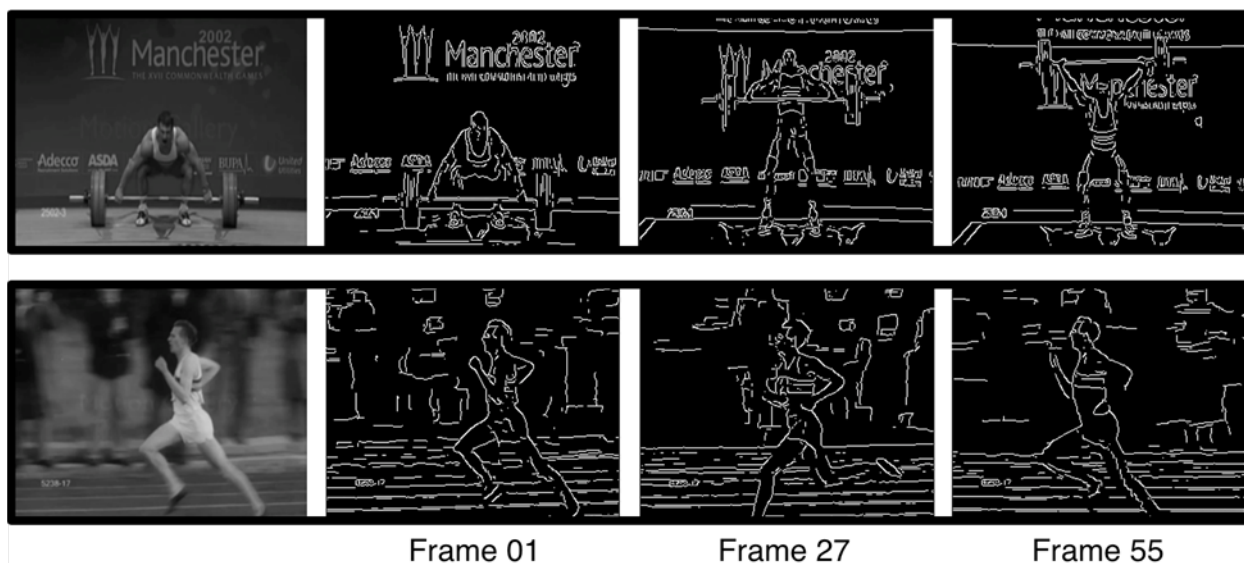


Figure 74: Final Binarized Stage 7 Output Following Application of Suppression Slope

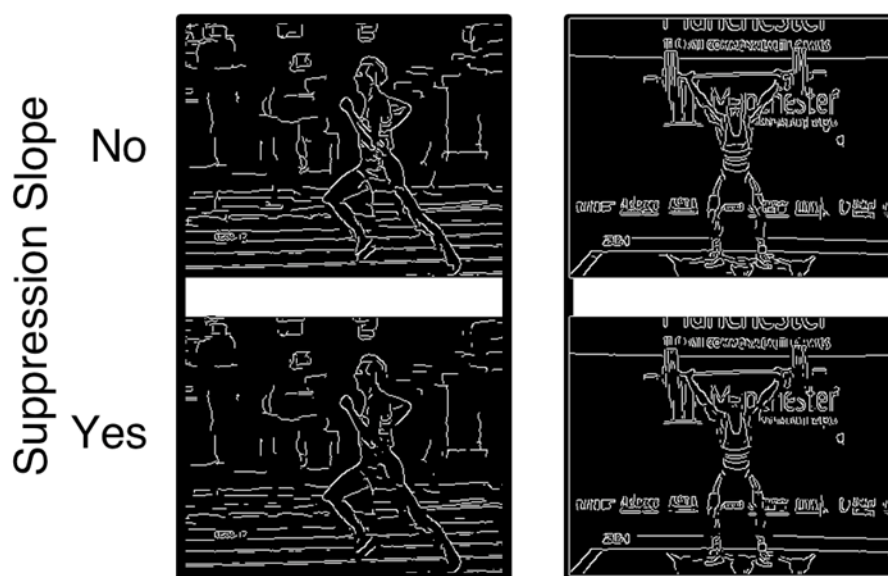


Figure 75: Comparison of Binarization Without (top) and With (bottom) Slope Suppression

8. Future Considerations

Edge detection using Gabor wavelets confers the advantage that edges in video can be detected at different spatial scales. We are currently investigating the merging of Gabors of different scales to further improve edge detection. We have also implemented and are testing a method to suppress jitter (e.g., sudden appearance/disappearance of contours) using spatiotemporal video characteristics. The method involves: detecting optic flow between frames in the grayscale video, segmenting regions that have supra-threshold motion magnitudes, preserving the sharpness of the moving regions, and iteratively blurring sub-threshold pixels across time.

We have also begun development of preprocessing that will isolate actions being performed in the scene by motion segmentation (i.e., delete background pixels). This spatiotemporal filtering helps reduce temporal noise in the edge maps the most in scenes with a stationary camera. We plan to develop this preprocessing method as a separate motion information channel in analogy to the *magnocellular* (motion) pathway of LGN. Once integrated, the model will have the edge channel (analogous to the *parvocellular* LGN pathway, which responds to more static and finer details), which is in spatial register with the motion channel. The combined edge and motion information should enhance the model's ability to learn higher-level, more non-linear, spatiotemporal feature/action/event categories. Finally, we emphasize that all of the preprocessing methods we are using are strictly local in space and time and: a) are not computationally expensive; and b) have substantial neurophysiological support.

APPENDIX B - SISC Property

Table 19 describes the experiments we conducted to establish the basis SISC property.

Table 19. Experimental Studies Described in this Section

Study	Brief Description
1	Basic SISC: Show that intersection between learned SDR codes co-varies with spatiotemporal similarity of the corresponding moments
2	Show how correlation between input moments and learned codes varies as function of the parameters of the sigmoid V -to- μ mapping (nonlinear neuronal activation function)
3	Demonstrate the generality of the SISC property/mechanism with a set of longer sequences. Specifically, show that the model intrinsically and automatically computes a spatiotemporal similarity metric that is sensitive to <i>all</i> time steps (moments) from start of sequence. We say that the temporal order of the spatiotemporal similarity metric is not of fixed order.

B.1 Study 1: Basic SISC Properties

The purpose of Study 1 is to demonstrate that TEMECOR maps spatiotemporally more similar inputs to more highly intersecting internal representations, i.e., SDR codes, which we refer to as the *similar-inputs-to-similar-codes* (SISC) property. This is an instance of what others have referred to as the “smoothness prior” (Bengio, Courville et al. 2012). The set of six 2-frame sequences used in this study are shown in Figure 76a; they are S0 through S5. All sequences have the same second item, X, while the pixel-wise overlap of the sequence-initial item with S0’s first item, A, decreases across sequences, S1=[BX], S2=[CX], etc. Thus, the spatiotemporal similarity of the second frame of each sequence with the second frame of S0 drops across sequences (even though the *purely spatial* similarity of the second frame remains the same at 100%). We will show that the codes assigned to the second frame of the progressively spatiotemporally less similar sequences have progressively smaller intersection with the code assigned to the second frame of S0. That code, for X, can be seen at right of Figure 76b.

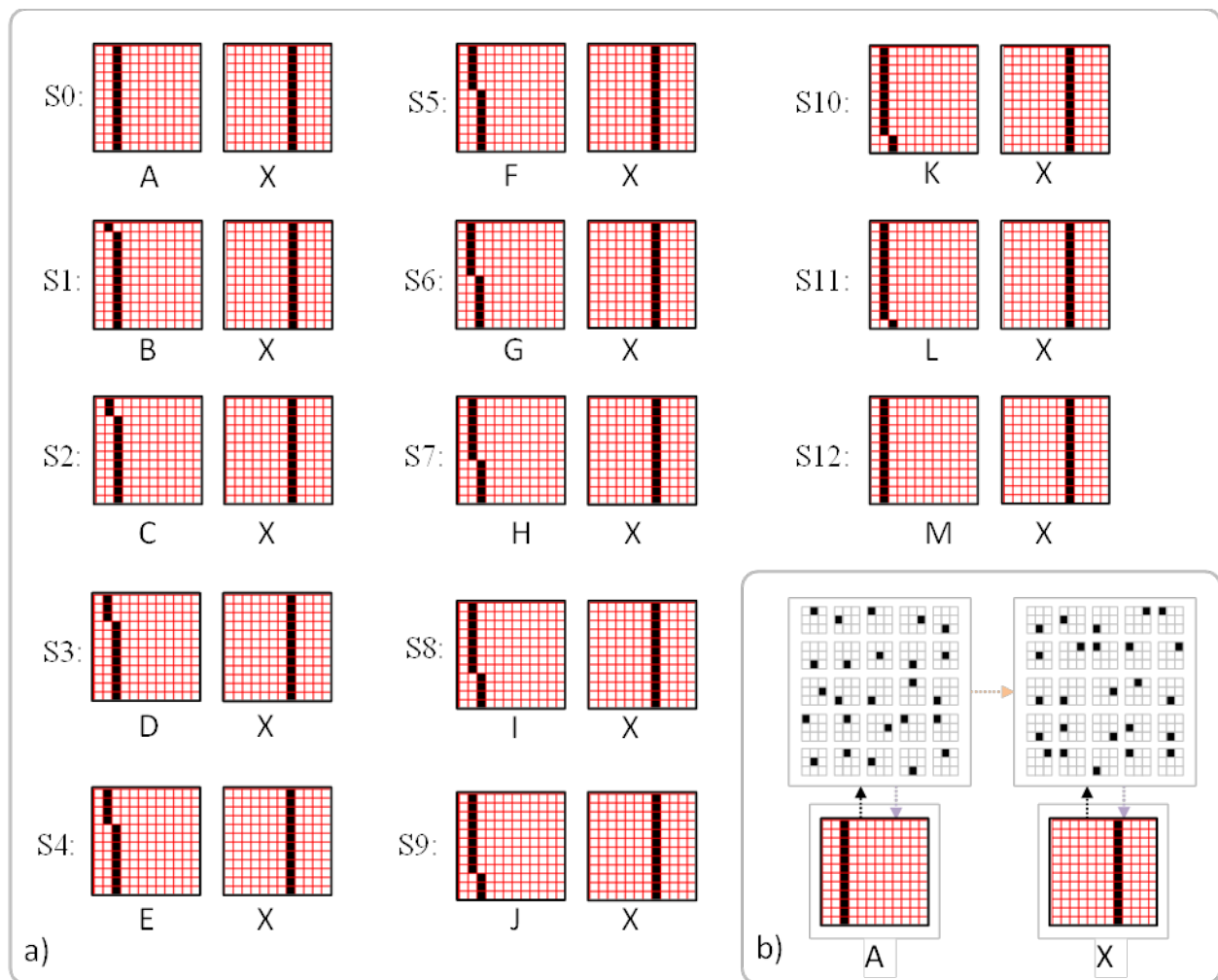


Figure 76: Six 2-Frame Sequences Used in Study 1 and Codes Assigned to S0

The model instance used in this study has two levels, an input level, L0, and one internal (representation) level, L1. L0 is a 12x12 pixel grid. L1 consists of a single macrocolumn (“mac”) consisting of $Q=25$ WTA competitive modules (CMs), a.k.a. “clusters” or minicolumns. Each CM consists of $K=9$ binary units (neurons). During learning, on each frame of an input sequence, an L1 code is chosen using TEMECOR’s Code Selection Algorithm (CSA) and associative learning occurs. Specifically, bottom-up (U) weights from active L0 units (active pixels) to active L1 units are increased to their maximum value, 1. Also, from the second frame ($T=1$) onward, horizontal (H) weights from L1 units active on the prior frame to currently active units are increased to 1. Figure 76b shows the memory trace assigned to sequence $S0=[AX]$. The trace consists of two codes, each consisting of 25 units, one unit in each of the 25 CMs. One might also refer to the set of weight increases made during presentation of [AX] as the “memory trace”, however, it is the sets of co-active coding units at each time step which, unless otherwise stated, we refer to as the memory trace of a sequence. Note that because [AX] is the first sequence presented to the model, the particular units chosen on both frames of S0 are chosen at random.

Before describing the results, we first describe our “moment” terminology. We refer to the frames of a sequence as representing particular “moments” in time. Thus, sequence S0 consists of two moments, the presentation of item A as the first item of the sequence and the presentation of X as the second item of the sequence when A was the first. To uniquely specify a moment, it is necessary to specify the full sequence (prefix) of items leading up to the moment. To distinguish a *sequence*, e.g., [ABCD], from a particular moment of that sequence, e.g., the third moment, when item C is presenting, we bold the item presenting at that moment, e.g., [ABC]. Thus, sequence [ABCD] consists of the four moments, [A], [AB], [ABC], and [ABCD]. Each moment of a sequence is mapped to a unique, *purely spatial*, L1 code; thus, codes of different Markov orders (different lengths of prefix on which the code depends) are stored in superposition in a mac. To foreshadow, this further implies that during retrieval, when the goal is to (re)activate the code of the most likely moment, all stored codes, reflecting moments of a wide range of Markov orders, all compete with each other. The CSA *effectively* searches over *all* stored codes, although crucially, *it does not iterate over* the stored codes. Rather, it iterates over the underlying representational units and the model’s weights, the numbers of which remain constant as additional codes are stored, conferring constant [a.k.a. “O(1)”] time complexity nearest-neighbor retrieval.

Figure 77 now shows, in panels b-f, the memory traces assigned to five sequences, [BX], [CX], [DX], [EX], and [FX], which are progressively less spatiotemporally similar to [AX]. In addition, Figure 77a shows the memory trace reactivated in response to a second presentation of [AX]. For each of the experiments represented by the six panels of Figure 77, the sequence shown is presented as the second sequence experienced by the model. For example, when S4=[EX] is presented, it is presented to the model after the model has only learned [AX], *not* the rest of the intervening sequences, S1-S3.

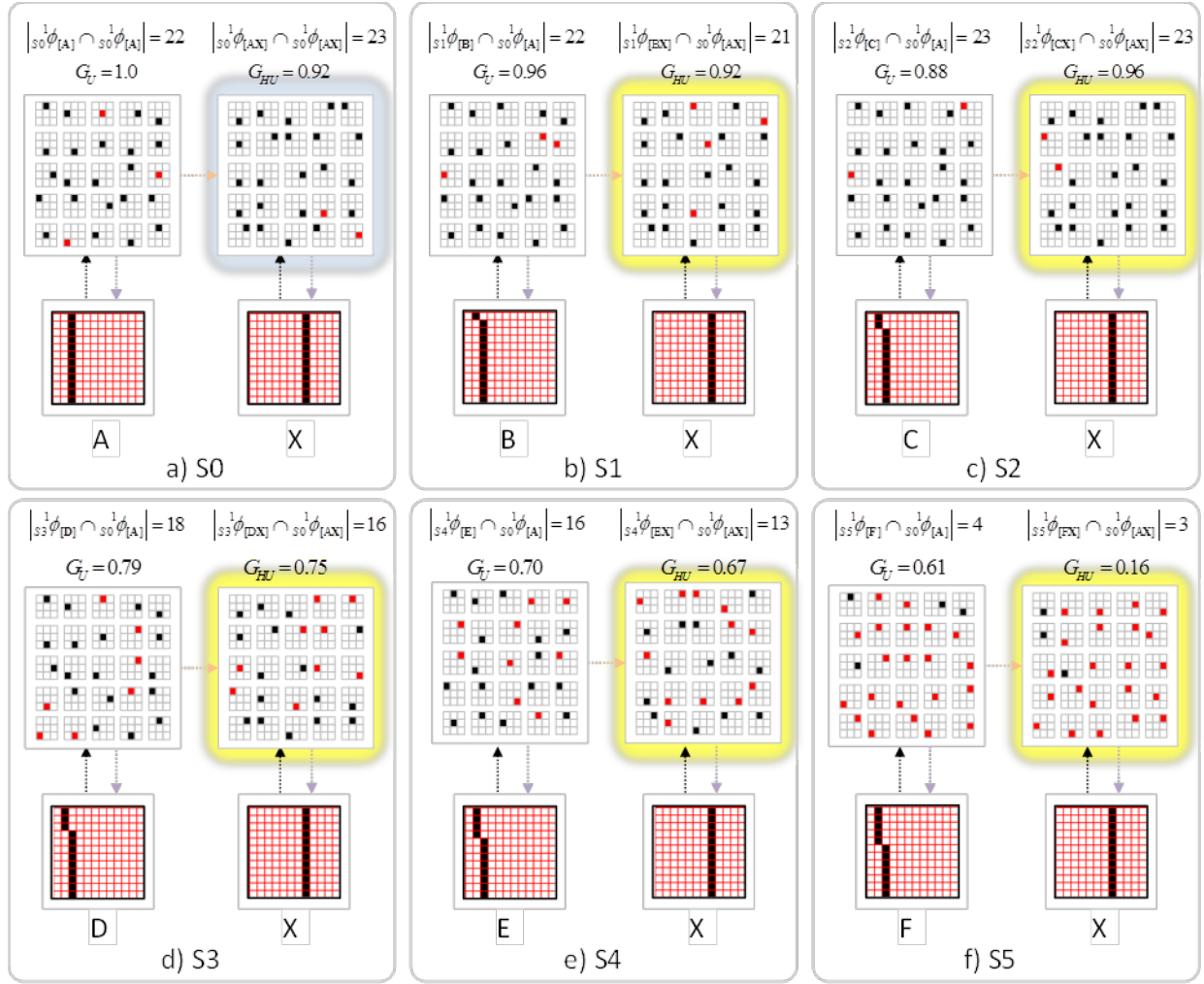


Figure 77: Portrayal of the Spatiotemporal SISC Property

The main result visible in Figure 77 is that in comparing the L1 codes assigned to frame 2 of each sequence, S1 to S5, to the L1 code assigned to frame 2 of S0 (in Figure 77b), we see progressively smaller intersection, i.e., the SISC property. These five L1 codes are highlighted in yellow and these are to be compared to the L1 code assigned to the second moment of S0 (highlighted in blue), [AX] (Figure 76b). Black units are units which are the same as for frame 2 of [AX] in Figure 76b; red units are different.¹³ The G values are the model's estimates of spatiotemporal similarity of the current moment. Thus, on the second moment, [BX], of sequence S1, the code assigned, ${}_{S1}^1\phi_{[BX]}$, has 21 out of the max possible 25 units in common with the code, ${}_{S0}^1\phi_{[AX]}$, assigned to the second moment, [AX], of S0, i.e., ${}_{S1}^1\phi_{[BX]} \cap {}_{S0}^1\phi_{[AX]} = 21$. The

¹³ If we viewed the presentations of S1 to S5 as recognition trials in which we were presenting progressively more perturbed variants of [AX], then these red units would be considered errors. However, in this case, we are viewing these presentations as presentations of similar but not identical sequences to S0, in which case it is appropriate for the model to assign unique codes. In this case, the red units are not errors, but simply just different from the unit chosen in the corresponding CM in frame 2 of S0.

code name convention here is that ϕ denotes a code, the superscript “1” indicates the model level at which code resides, i.e., L1, the lead subscript indicates the sequence in which the code occurs, and the trailing subscript, the specific moment of the sequence that the code represents. As the spatiotemporal similarity of the second sequence moment with [AX] decreases further across panels c-f, the intersection of the assigned code with ${}_{s0}^1\phi_{[AX]}$ trends downward, despite the fact that in this particular instance, ${}_{s1}^1\phi_{[CX]} \cap {}_{s0}^1\phi_{[AX]} = 23$ even though [CX] must clearly be considered less similar to [AX] than [BX] is to [AX]. Despite this statistical blip, the codes assigned for the remaining progressively less spatiotemporally similar moments, [DX], [EX], and [FX], have monotonically decreasing intersection with ${}_{s0}^1\phi_{[AX]}$ as summarized in the right-hand column of Table 20 below. In fact, the same trend obtains with respect to the first sequence moment as well (left-hand column). However, note that in the latter case, it is purely spatial similarity in the input space that is relevant (since no temporal context information is present on the first moment of a sequence).

Table 20. Code Similarity Decreases with Spatiotemporal Similarity of Moments

Decreasing Similarity of 1 st Moment	Decreasing Similarity of 2 nd Moment
${}_{s0}^1\phi_{[A]} \cap {}_{s0}^1\phi_{[A]} = 22 \quad (88\%)$	${}_{s0}^1\phi_{[AX]} \cap {}_{s0}^1\phi_{[AX]} = 23 \quad (92\%)$
${}_{s1}^1\phi_{[B]} \cap {}_{s0}^1\phi_{[A]} = 22 \quad (88\%)$	${}_{s1}^1\phi_{[BX]} \cap {}_{s0}^1\phi_{[AX]} = 21 \quad (84\%)$
${}_{s2}^1\phi_{[C]} \cap {}_{s0}^1\phi_{[A]} = 23 \quad (92\%)$	${}_{s2}^1\phi_{[CX]} \cap {}_{s0}^1\phi_{[AX]} = 23 \quad (92\%)$
${}_{s3}^1\phi_{[D]} \cap {}_{s0}^1\phi_{[A]} = 18 \quad (72\%)$	${}_{s3}^1\phi_{[DX]} \cap {}_{s0}^1\phi_{[AX]} = 16 \quad (64\%)$
${}_{s4}^1\phi_{[E]} \cap {}_{s0}^1\phi_{[A]} = 16 \quad (64\%)$	${}_{s4}^1\phi_{[EX]} \cap {}_{s0}^1\phi_{[AX]} = 13 \quad (52\%)$
${}_{s5}^1\phi_{[F]} \cap {}_{s0}^1\phi_{[A]} = 4 \quad (16\%)$	${}_{s5}^1\phi_{[FX]} \cap {}_{s0}^1\phi_{[AX]} = 3 \quad (12\%) (\sim\text{chance})$

We emphasize that each of the memory traces shown in Figure 77 is a particular instance. The winner in a CM is chosen as a draw from a likelihood distribution over the CM’s units, i.e., “softmax”, *not* by simply choosing the max likelihood unit, i.e., plain (“hard”) max. Thus, we will generally see some variation in the chosen codes across instances of the same experiment and the amount of variation will increase as the similarity of the test sequence to the learned sequence, [AX], decreases. This statistical variation, for example, is why the memory trace in Figure 77a is not perfect. Due to the statistical nature of Sparsey’s CSA, demonstration of the SISC property requires running many instances of each of the experiments shown in Figure 77 and reporting average results. Such a protocol was followed in Study 2. However, before moving to Study 2, we present a detailed explanation of how the probabilistic CSA works and how it realizes the SISC property. This is partially redundant to Section 3.2, however it elaborates the explanation substantially, particularly with respect to achieving SISC.

B.2 Explanation of the Code Selection Algorithm (CSA)

Important Note: The version of the CSA stated in Table 21 is an older, less detailed version than stated in Table 2. Further, it differs notationally. In particular, the sigmoid parameters have different identifiers, as does the un-normalized win probability, which is denoted in the section about was denoted throughout the rest of the report. We decided to stick with these alternate conventions because this section is self-contained and contains many figures and notations that would have had to have been updated.

Table 21. The Code Selection Algorithm

	Equation	Short Description
1a	$u(i) = \sum_{j \in RF_U(t)} w(j, i)$	Compute unit i 's raw U, H, and D input summations, $u(i)$, $h(i)$, and $d(i)$, independently. $RF_U(t)$ is the set of units in i 's bottom-up <i>receptive field</i> , RF_U , which are active at time, t , $RF_H(t-1)$ is the set of active units in i 's horizontal RF, RF_H , active at $t-1$, and $RF_D(t)$ is the set of units in i 's top-down RF, RF_D , active at t . Again, because unit activations are binary, we can simply sum the weights, $w(j, i)$, which are also binary.
1b	$h(i) = \sum_{j \in RF_H(t-1)} w(j, i)$	
1c	$d(i) = \sum_{j \in RF_D(t)} w(j, i)$	
2a	$U(i) = \begin{cases} u(i)/S & L = 1 \\ u(i)/f_U(RF_U) & L \geq 2 \end{cases}$	Normalize the summations independently. In the case of the U input to L1, we normalize by the expected number of active units (pixels) in an L1 mac's RF_U , which we denote by S . For cases in which the afferent field is organized into macs, normalization is more complex. These f functions are placeholders and will be specified/discussed as needed.
2b	$H(i) = h(i)/f_H(RF_H)$	
2c	$D(i) = d(i)/f_D(RF_D)$	
3a 3b	$V(i) = \begin{cases} H(i)^{\lambda_H(t)} U(i)^{\lambda_U(t)} D(i)^{\lambda_D(t)} & t \geq 1 \\ U(i)^{\lambda_U(t)} & t = 0 \end{cases}$	Compute local evidential support. On the 1 st time step ($t=0$), before codes have become active throughout the hierarchy, only U signals are available at each level. Thereafter, all signals are present at each level and are multiplicatively combined. The exponents modulate the sensitivity to the similarity of each source, and thus, the shape/granularities of the spatiotemporal tuning functions learned.
4	$\hat{V}_j = \max_{i \in C_j} \{V(i)\}$	Find the max V , \hat{V}_j , in each CM, C_j .
5	$G \equiv \sum_{q=0}^{Q-1} \hat{V}_k / Q$	Compute G as the average \hat{V} value over the Q CMs. G is a population-level measure of the spatiotemporal familiarity of the current moment.
6	$\eta = K \times \pi \times \left(\left[\frac{G - G_T}{1 - G_T} \right]^+ \right)^\chi$	Determine the range, η , of the of the sigmoid activation function, which transforms a unit's V value into its relative (within its own CM) probability of winning, μ , the "V-to- μ map". In a more general development, this step may also determine any of the other sigmoid parameters, α , β , γ and ν , which interact to control overall sigmoid shape. However, in this report, α , β , γ , and ν will be fixed on any particular run.
7	$\mu(i) = \frac{(\eta - 1)}{(1 + \gamma e^{-\alpha(V(i) - \beta)})^\nu} + 1$	Apply sigmoid activation function to each unit. Note: the sigmoid collapses to constant function, $\mu(i)=1$, when $\eta=0$ (i.e., when $G \leq G_T$).
8	$\rho(i) = \frac{\mu(i)}{\sum_{j \in CM} \mu(j)}$	In each CM, normalize the relative probabilities of winning (μ) to final probabilities (ρ) of winning.
9	Select a final winner in each CM according to the ρ distribution in that CM, i.e., soft max.	

We will describe the steps of Table 21 and via a series of examples to follow. At the most general level, the CSA chooses units to participate in codes in *probabilistic* fashion, i.e., as draws from distributions (Step 9). Broadly the CSA modulates the distributions (in the CMs), on a moment-to-moment basis, with the goal of increasing the probability that the *locally* most favored unit, i.e., the one with the highest V in its CM, wins in proportion to the *global spatiotemporal familiarity*, G , of the current input moment. Provided the amount of information—more specifically, the number of codes—stored in a mac is low enough so that the effects of crosstalk (interference) between codes remain sufficiently small, G approximates the maximum *spatiotemporal similarity* of the current moment to all previously experienced (and thus, stored) moments. The ultimate effect of this distribution modulation policy is that more spatiotemporally similar moments get assigned to more highly intersecting codes, i.e., the SISC property. In this report, the primary means of modulating the distributions will be to vary the range, η , of the sigmoid activation function (Step 6) through which a unit's V value is passed (Step 7) on its way to determining a unit's final probability (ρ) of being chosen (Step 8). With this broad overview in mind, we now proceed to a series of detailed examples of the CSA in operation on some of the moments of the sequences in Study 1.

Figure 78 shows how the distributions from which winners are ultimately drawn are computed by the successive stages of the CSA on presentation of A, given that sequence S0=[AX] has previously been presented and learned. The figure shows the details for only a few CMs, though conditions in all $Q=25$ CMs are statistically the same. Broadly, the U signals arising from the 12 active pixels of input A give rise to maximal u -summations for the units which won and were assigned as the code, ${}_{S0}^1\phi_{[A]}$, of input A when it first occurred (see Figure 78b). The U connections carrying these signals (blue) are shown for a few of the L1 units. The maximal u -summations normalize to U -values of 1 and, via the rest of the CSA (working up the rows of the panel b), to final win probabilities (ρ) of 92%. Because there is no crosstalk between stored codes in this example, each of the other eight units in each CM have $u=0$, $U=0$, and ultimately ~1% chance of winning (ρ). In the particular run depicted in Figure 78a, it so happens that in three of the $Q=25$ CMs, e.g., CM 2, one of these other units (red) was drawn from the distribution. Thus, even though the input moment [A] is 100% familiar, the CSA's probabilistic nature, allows less likely units to be chosen occasionally.

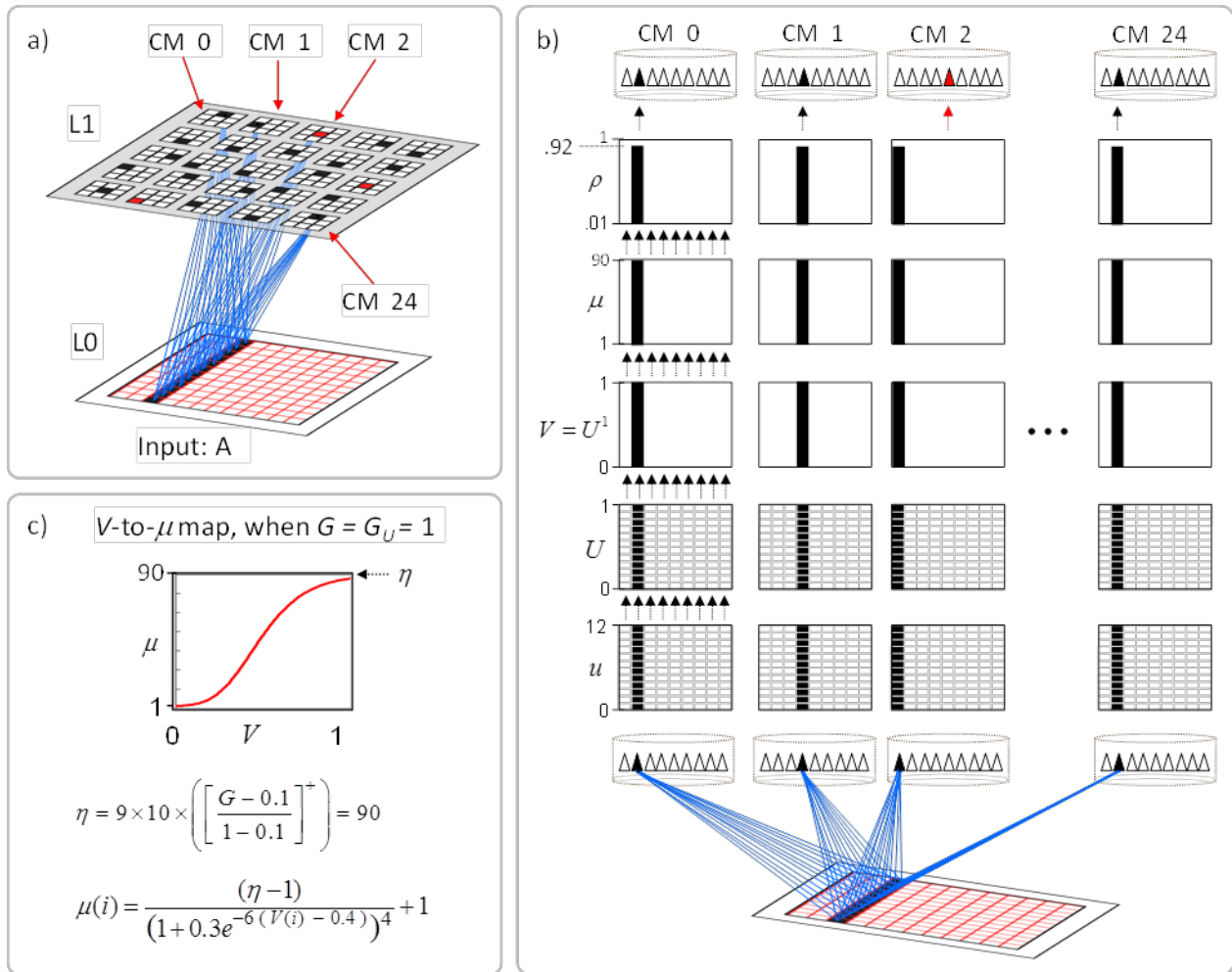


Figure 78: Graphical Explanation of How Winners are Chosen in the CMs and Thus, How Entire Codes are Chosen

The reader may ask why the model does not simply pick the unit with maximum V in each CM. In this particular case, that would ensure perfect reactivation of ${}_{s_0}^1\phi_{[A]}$. Why allow the other eight units in each CM to have any chance of winning? The reason is that in general, when an intelligent system, natural or machine, is operating autonomously in a real environment, it cannot know *a priori*, whether any particular moment that it experiences will be novel or familiar. In other words, it cannot know in general whether it is or should be in learning (train) or retrieval (test) mode on any particular moment. It must be able to learn potentially new information at every moment. Thus, while it is true that when the model is presented with an *exact* duplicate of an earlier moment (as in the case of Figure 78), the optimal policy is simply picking the max- V unit in each CM, that is not the case when the moment is less than 100% familiar. If the current moment contains any novelty (new information) at all, it could be that that new information is important, in which case, the model must assign a *new* code for the moment. That is the reason for the CSA's policy of *drawing* winners from distributions rather than simply picking the unit with maximum support, V .

Figure 78c describes the transformation from V values (which reflect only information that is purely *local* to a unit) to μ values (which reflect information *global* to the entire mac), the “ V -to- μ map”. Specifically, that global information is the aforementioned global spatiotemporal familiarity of the current input moment, G . Formally, G is just the average of the maximum V values in the CMs (CSA Step 5). Thus, G is normalized to $[0,1]$. The reasoning behind the definition of G is given in Table 22 below.

Table 22. Reasoning Underlying the Global Spatiotemporal Familiarity Measure, G

1	Assume that the current moment ξ has been experienced before.
2	Then it will have been assigned a code ${}^1\phi_\xi$ and the weights from all active presynaptic units to the units comprising ${}^1\phi_\xi$ will have been increased to the max weight of 1.
3	By 1 and 2, the set of currently active presynaptic units is the same as in the earlier instance of ξ .
4	By 2 and 3, all of the units comprising ${}^1\phi_\xi$ will have $V=1$.
5	Thus $G = \sum_{q=0}^{Q-1} \hat{V}_q / Q = 1$, indicating complete <i>familiarity</i> of the current moment.
6	On the other hand, assume that the current moment ξ differs from all previously experienced moments.
7	In this case, the average V value over the units comprising any specific previously assigned code ${}^1\phi_\zeta$, which we denote $\bar{V}_\xi({}^1\phi_\zeta)$, will be less than 1, <i>provided the number of codes have been stored is low enough so that the effects of crosstalk (interference) between codes remain sufficiently small</i> .
8	Again, provided crosstalk is sufficiently low, Point 7 implies that the expected G value will also be less than 1, indicating the lower familiarity of the current moment.
9	However, the strength of the implication from 7 to 8 diminishes and is eventually nullified as crosstalk increases. Thus, Sparsey requires a mechanism to keep crosstalk below a threshold. That mechanism is to prevent further learning once a specified fraction of a unit’s afferent weights have been increased, i.e., impose a “critical period”.

In accord with the fact that Sparsey’s codes are sparse distributed codes, i.e., *sets* of co-active units, which are assigned as wholes (i.e., in unitary events), it is appropriate that the familiarity measure used by the model, G , is a population-, or code-level (i.e., global) measure, not a unit-level (local), measure.

In CSA Step 6, G is then used to set the range, η , of the sigmoid activation function. η is maximized when $G=1$. In this example, the number of units per CM, K , equals 9. We refer to the parameter, π , as the *sigmoid expansion factor* and parameter, χ , as the *sigmoid expansion*

exponent. To simplify explanation in these studies, χ is set to 1. The value $\pi=10$ in this Study was chosen arbitrarily, but π 's effect on the relative (μ) and total (ρ) probability distributions will quickly be made clear in the rest of this section. The larger is π , the greater is η , and the higher the probability that the unit with the maximum V value in a CM wins. The particular ρ values in Figure 78b, i.e., 92% for the units with $V=1$ and 1% for units with $V=0$, depend directly on the choice of $\pi=10$. We will demonstrate the effect of varying π shortly, but first we continue our thread of explaining how the SISC property is achieved (for any particular value of π).

Figure 79 describes the probabilistic code selection on the second moment, $[AX]$, of the second presentation of $S0$. This figure is quite similar to the previous one except that it includes the horizontal (H) signals, which arise from the L1 code activated on the prior time step (moment), $[A]$, and arrive back at this same field of units on the second moment, when X is the input (a representative sample of these H signals are shown as blue arcs from the previous moment's winners (gray) to the current winner in CM 2). On the second and subsequent time steps of any sequence, a unit's local evidential support includes this temporal context information and is in fact simply computed as the product of its normalized inputs. In this case, since level L1 is the top level, there are no D signals. Thus, CSA Step 3a simplifies to just the product of the H and U factors. The logic behind determining a unit's total support, V , in this way is given in Table 23 below.

Table 23. Reasoning Underlying the Computation of a Unit's Local Support, V

1	If a unit, x , was chosen winner and thus included in a code for some moment ξ in the past, then its afferent synapses, U , H , and D (if present), from <i>all</i> units active on that occasion will have been increased to 1.
2	In fact, x may generally have been included in the codes of many past moments. If <i>any</i> of those moments is identical to the current moment, then x 's current input summations will be maximal and its U , H , and D (if present) values will equal 1. In this case, multiplying these normalized summations yields a value of $V=1$, correctly indicating that x has maximal local evidential support.
3	On the other hand, to the extent that any of U , H , or D fall below 1, that indicates decreasing spatiotemporal similarity of the current moment to any past moment, or in other words, increasing novelty of the current moment. In this case, the product of U , H , and D , will reflect this.

A unit's normalized u value, U , is computed by dividing u by the size of input patterns, i.e., the number, S , of active pixels, which in these examples is held constant at $S=12$. The normalized h value, H , is computed by dividing h by then number of afferent H-signals that a unit expects to have. In this model instance, the H matrix is nearly full; specifically, all L1 units have H synapses onto all other L1 units except those in their own CMs. Since all L1 codes consist of exactly $Q=25$ active units, it follows that the maximal h a unit can have is 24. Thus, CSA Step 2b becomes: $H(i) = h(i)/24$.

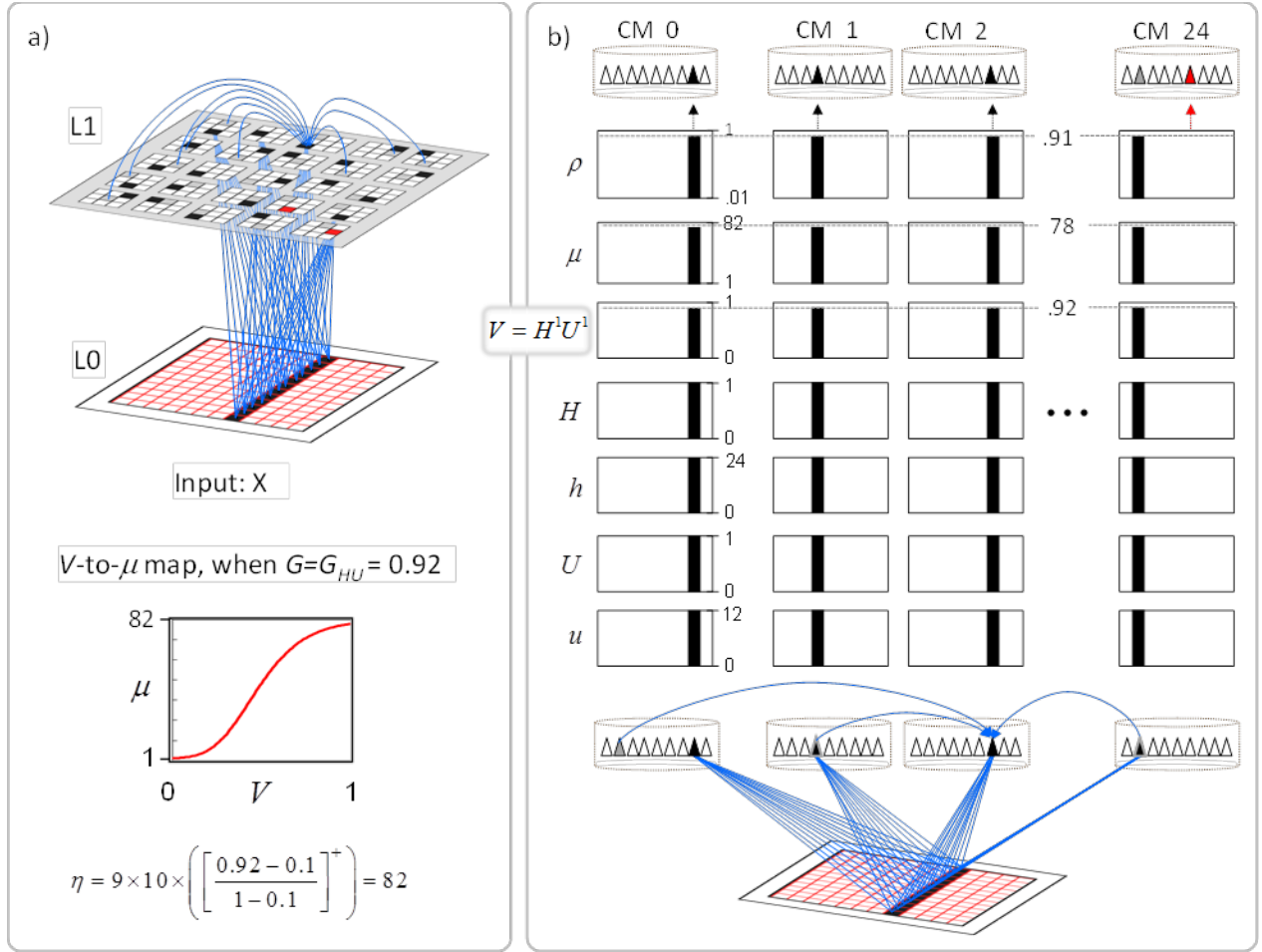


Figure 79: Graphical Explanation of Code Selection on the Second Moment, [AX], of the Second Presentation of S0

As external observers to the system, we know that this second moment [AX] is identical to the one experienced during the first presentation of [AX] (in Figure 76b). However, all that the model has to help judge the spatiotemporal familiarity of the current moment is the information coming from its internal state at the prior moment, i.e., the H-signals arriving from the previously active L1 code, ${}_{S0}^1\phi_{[A]}$. Because of the three incorrect units (red) activated as the code for the prior moment [A] (in Figure 77a and Figure 78a), the h -summations for the units comprising the code originally assigned to [AX] (in Figure 76b), will be lower than expected. When normalized, these h -summations yield $H \approx 0.92$ for those units. By CSA Step 3a, these units also have $V \approx 0.92$. This yields $G_{HU} = 0.92$, which in turn yields $\eta = 82$. In CSA Steps 7 and 8, the units' V values are transformed to relative (μ) and then to total (ρ) probabilities. In this case, the correct unit in each CM ends up with ~91% chance of winning and the other eight units, with ~1% chance of winning. Finally in CSA Step 9, winners are chosen from those distributions. In this case, the wrong unit (red) is chosen in 2 out of $Q=25$ CMs. In other words, the L1 code for this second instance of moment [AX] has very high overlap with the L1 code chosen in that initial experience of [AX].

Figure 80 shows the situation when we present $S3=[EX]$ after the model has learned $[AX]$. In this case, items E and A have only 8 out of 12 features (pixels) in common. This leads to u -summations of 8 for each of the 25 units that won when A was presented, i.e., the 25 units comprising the code, ${}_{S0}^1\phi_{[A]}$. With Figure 80, we begin to see how the CSA realizes the SISC property. The lower u values yield $U=0.79$ for the units of ${}_{S0}^1\phi_{[A]}$. By CSA Steps 3-5, this yields $G=0.7$, which by CSA Step 6, yields a maximal μ value, $\eta = 60$. In Step 7, the V values of all units are passed through the sigmoid activation function. In this case, the units that were in the code, ${}_{S0}^1\phi_{[A]}$, have $V=0.7$, which maps to $\mu \approx 28$. Renormalization of the μ values in Step 8 yields final win probabilities of $\rho = 77\%$. The other eight units in each CM have $V = 0$, which maps to $\mu = 1$ and then to $\rho \approx 3\%$. These particular values depend on the particular sigmoid parameters used in this example, $\alpha = 14$, $\beta = 0.9$, $\gamma = 0.4$, $\nu = 0.4$. These values were chosen to give a long approximately linearly-increasing region, yielding a correspondingly linear correlation between G and code intersection size over an appreciable range of V . The ultimate result of the fact that the spatiotemporal similarity (as measured by G), $Sim([E],[A])$, is lower than $Sim([A],[A])$ is that $\left| {}_{S4}^1\phi_{[E]} \cap {}_{S0}^1\phi_{[A]} \right| = 16$ is lower than $\left| {}_{S0}^1\phi_{[A]} \cap {}_{S0}^1\phi_{[A]} \right| = 22$.

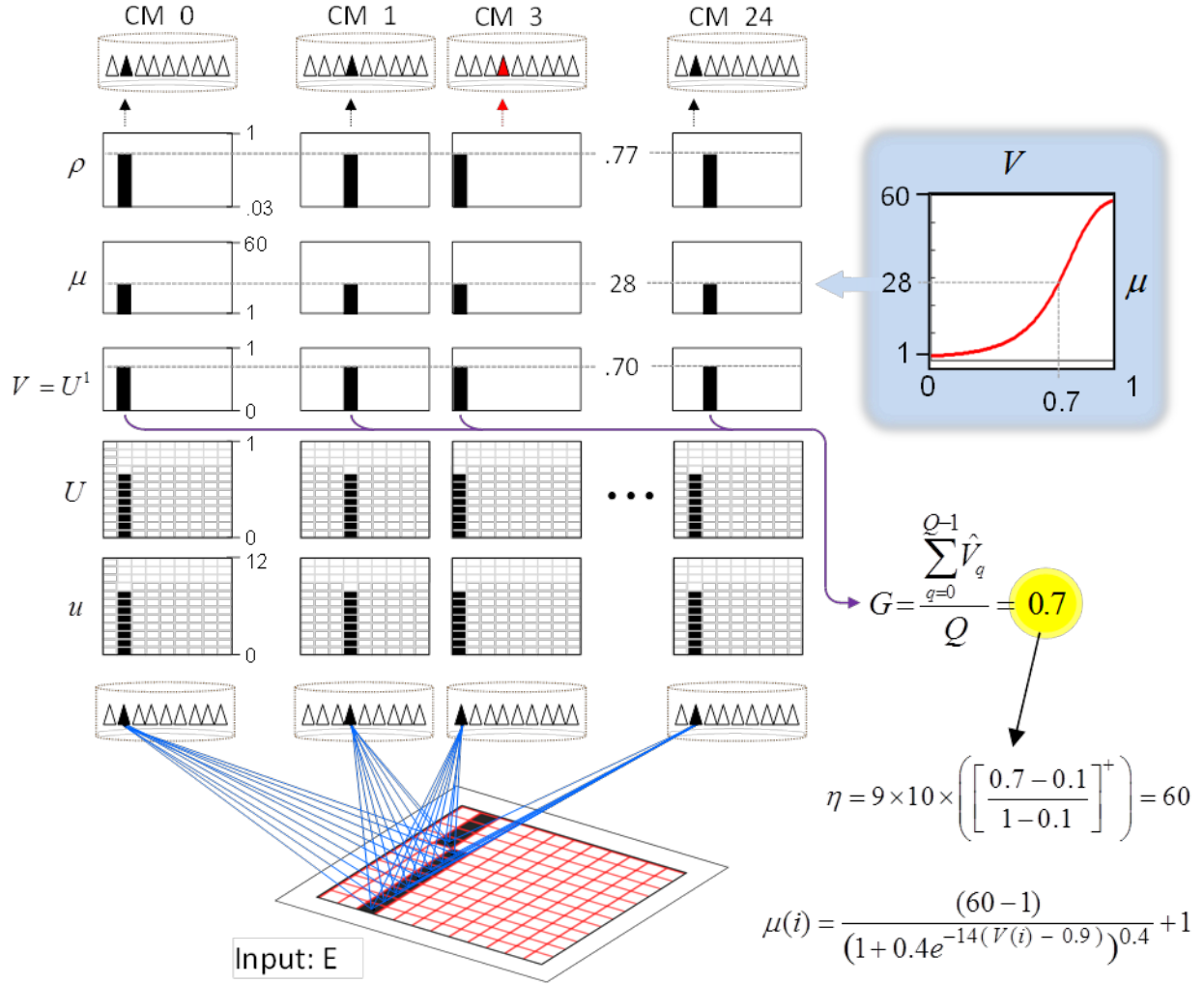


Figure 80: Graphical Explanation of Code Selection on First Moment, [E], of S3=[EX]

For completeness, Figure 81 shows the presentation of the second moment, [EX], of sequence [EX]. While the current bottom-up input, from item X, is perfectly familiar, this is the second moment and so the model computes a *spatiotemporal* familiarity based on multiplicatively combining the current U signals with the temporal context information carried in the H signals. The novelty of the prior moment, [E], causes the model to find the current moment quite novel (i.e., $G_{HU} = 0.7$), which yields compressed distributions ($\eta = 60$) and ultimately, choosing a unit other than the max-V unit (red units in Figure 81) in 12 of 25 CMs.

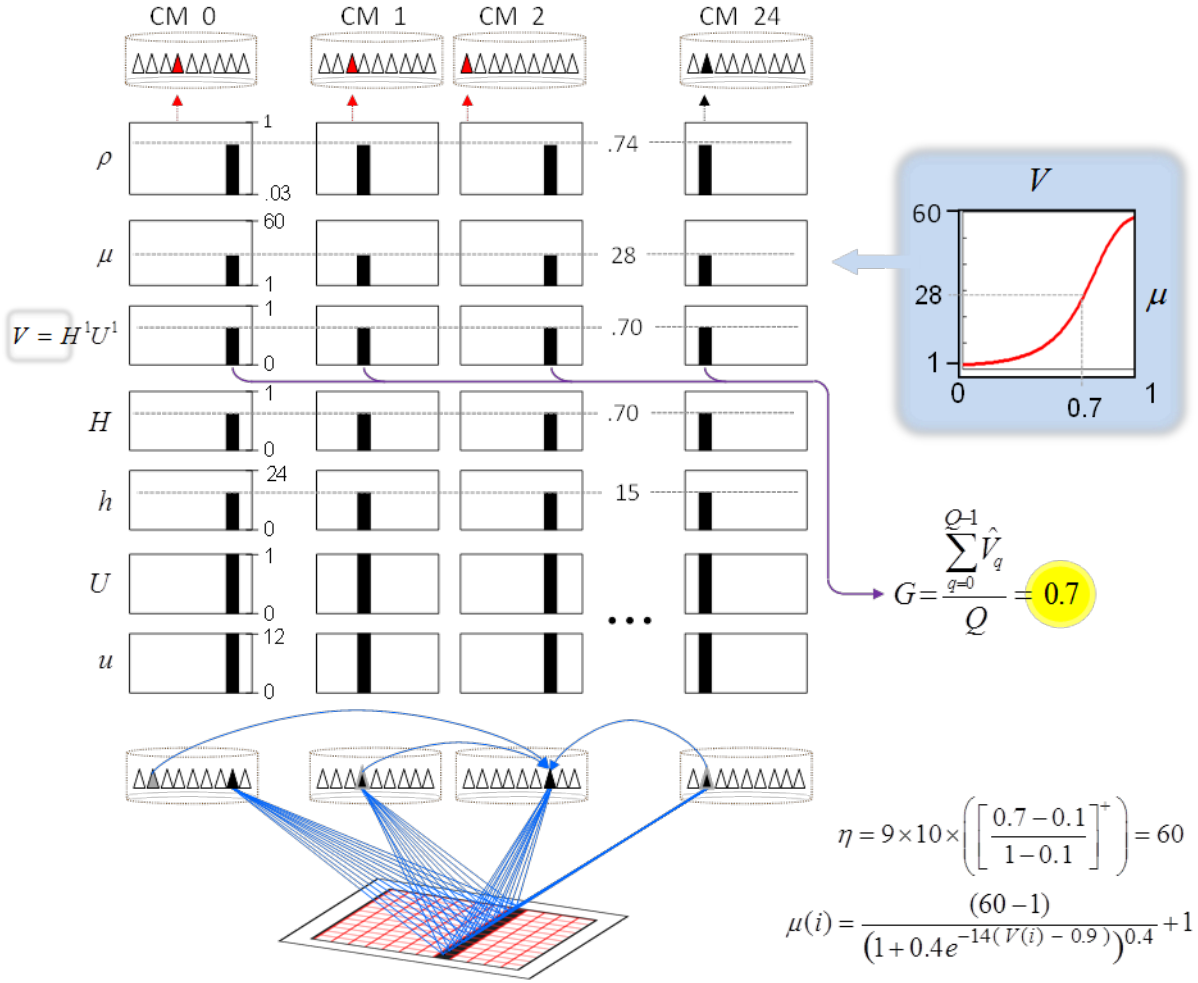


Figure 81: Graphical Explanation of Code Selection 2nd Moment, [EX], of S3=[EX]

B.3 Study 2: SISC Properties as Function of V-to-μ Mapping Parameter, π

The previous four figures have hopefully elucidated the essential nature of Sparsey's probabilistic code selection algorithm (CSA), showing why it achieves the SISC property. Again, these four examples referred to particular runs. However, to clearly demonstrate the correlation between spatiotemporal familiarity, G , and code intersection, we must report averages over many runs. We do so here in Study 2. Study 2 uses the same model trained in the learning phase of Experiment 1, i.e., the network of Study 1 trained on the single sequence, [AX]. The test set was the same as in Study 1 but included four additional sequences, S6=[GX], S7=[HX], S8=[IX], and S9=[JX]. All data points in Table 24 are averages of 30 runs.

We also promised earlier to demonstrate the effect of varying the parameter, π (the *sigmoid expansion factor*). Increasing π stretches the sigmoid, giving proportionately greater μ values to units with higher V values compared to those with lower V values. If a unit has a high V value for a given input moment, then it has been active in past moments that are spatiotemporally similar to the current moment. Thus, increasing the relative advantage of higher V units over lower V units makes units that have been included in the codes of more similar moments

relatively more likely to win than units that have been included in the codes of less similar moments. This means that, all else equal, increasing π leads to the formation of *broader* (larger) categories and decreasing it, to *finer* (smaller) categories.

This correlation can clearly be seen in Table 24. Rows correspond to different test sequences organized in decreasing similarity order. Column two gives the pixel-wise overlap of the first moment of the test sequence with the first moment, $[A]$, of S_0 . Code similarity is reported as intersection size with the code of the second moment, $[AX]$, of S_0 expressed as a percent of the maximum possible intersection size, i.e., $Q=25$. Again, all reported values are averages over 30 runs. Thus, the correlation between category broadness (coarseness) and π is clearly evident. For any given sequence (row), moving across the table to higher π values leads to higher intersections with ${}_{S_0}^1\phi_{[AX]}$. At $\pi=1000$, even very spatiotemporally different moments, e.g., $[JX]$, are mapped to ${}_{S_0}^1\phi_{[AX]}$. That is, at $\pi=1000$, all of these sequences are co-classified.

At the other extreme, when $\pi = 10$, we see a wide range of intersection sizes starting at 91% and dropping down to 16% (which corresponds approximately to chance level). In particular, for $\pi=10$, the correlation between spatiotemporal similarity of input moments and code intersection size, i.e., the SISC property, can be reasonably approximated as linear from sequence $[DX]$ through $[IX]$. This dynamic range of the SISC property shrinks as π increases.

Table 24. Variation of SISC Property with Parameter, π

		$ {}_{S_0}^1\phi_{[X]} \cap {}_{S_0}^1\phi_{[AX]} $ (as %)				
Seq.	Sim($[_],[A]$)	$\pi = 10$	$\pi = 20$	$\pi = 50$	$\pi = 100$	$\pi = 1000$
$[AX]$	12/12	91	94	98	99	100
$[BX]$	11/12	90	96	97	99	100
$[CX]$	10/12	89	96	99	99	100
$[DX]$	9/12	88	94	97	99	100
$[EX]$	8/12	87	94	98	99	100
$[FX]$	7/12	72	93	97	100	100
$[GX]$	6/12	52	83	97	99	100
$[HX]$	5/12	29	69	95	98	100
$[IX]$	4/12	17	37	81	96	100
$[JX]$	3/12	15	17	53	84	100

The preceding discussion considered varying π from the vantage point of learning as a way to control category granularity. However, we can also consider varying π from the vantage point of retrieval. In this case, we could view the sequences from $[BX]$ to $[JX]$ as progressively degraded instances of $[AX]$ in which case increasing π makes the model better able to retrieve the correct memory in the face of degradation/noise (i.e., “clean up memory” in the associative memory nomenclature). Specifically, increasing π increases the probability that the most favored (highest V) unit in each CM wins, and indirectly, the probability of the *whole code* of the most

spatiotemporally similar moment (in this case, ${}^1_{s0}\phi_{[AX]}$), or in other words, the most likely hypothesis, being activated.

Earlier, we pointed out that if the model “knows” that it is in a *retrieval* trial and will not be asked to learn anything new (even if the current input moment does contain new information), then the optimal strategy is simply to pick the max- V unit in each CM. In particular, this would cause 100% recognition of the second presentation of [AX] (in Figure 79). As can be readily extrapolated from Table 24, increasing π asymptotically achieves this same result; as $\pi \rightarrow \infty$, the CSA approaches simply choosing the hard max (of V) in each CM.

To aid intuition, we include Figure 82 which is essentially a repeat of the Figure 81, except that π has been set to 100. Even though the max- V unit in each CM still only has $V=0.7$, the greatly expanded μ range ($\eta=600$) greatly increases the probability of choosing these max- V units, $\rho > 97\%$. This higher probability (than when $\pi = 10$) substantially increases the probability that the entire code, ${}^1_{s0}\phi_{[AX]}$, is reactivated.

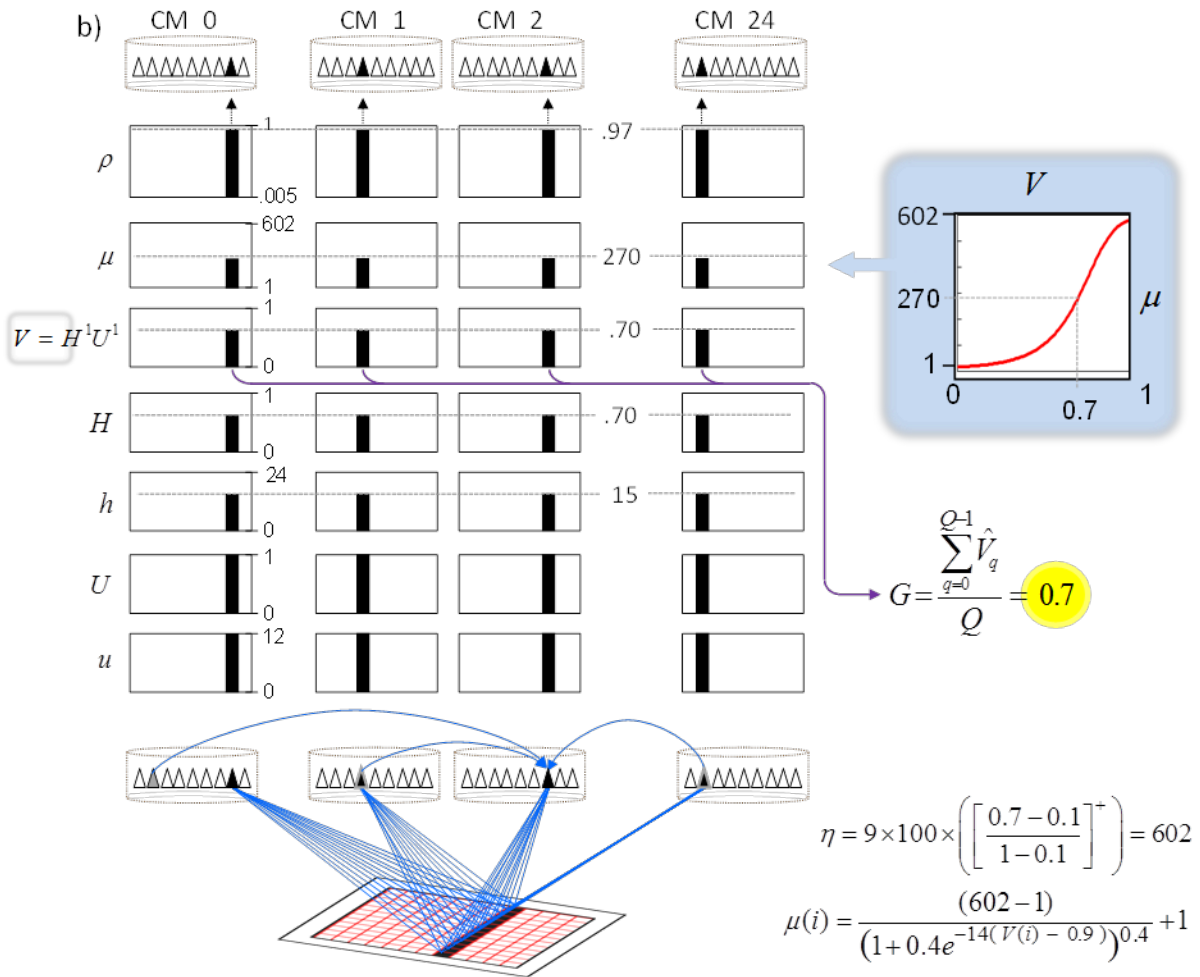


Figure 82: Presentation of Moment [EX] when $\pi = 100$

B.4 Study 3: Arbitrary Temporal Context Window Length of SISC Property

This study also uses a 2-level model. In the training phase, the 4-item sequence, [ABCD], Seq. 0 in Figure 83b, is presented once, resulting in a trained network. For each of the original frames, we constructed a sequence of three progressively less similar frames, e.g., reading down the leftmost column of Figure 83a, we see the original frame, A, with 12 active pixels, then A' which has 9 pixels in common with A, then A'' with 6 pixels in common with A, and finally A''' with only 3 pixels in common with A: similarly, for B, C, and D.

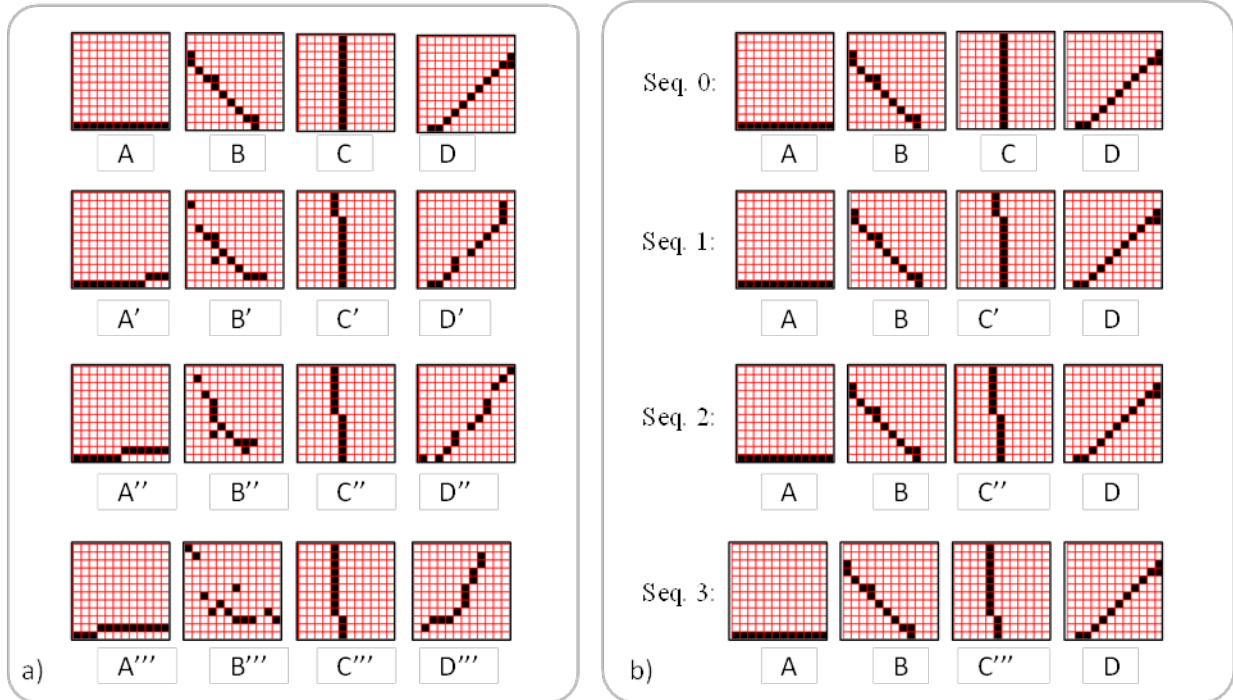


Figure 83: (a) Library of 16 Input Items from which Train and Test Sequences are Constructed and (b) The Input Set for Study 3

We then created a large number of 4-frame test sequences from this library, which are listed in Figure 84. The blocks in Figure 84 are organized so that the spatiotemporal similarity with the original sequence, [ABCD], decreases with row. For example, Sequences 1-3 of Figure 83b (which correspond to the block “Exp. 1”) clearly decrease in similarity with [ABCD] since the third moment of each sequence, i.e., C', C'', and C''', decrease in spatial similarity to C, while the spatial inputs at the other three moments, i.e., A, B, and D, remain constant across the sequences. The goal of this study is to demonstrate that the model’s SISC property is sensitive to *all* moments, from the start of the sequence, leading up to the current moment.

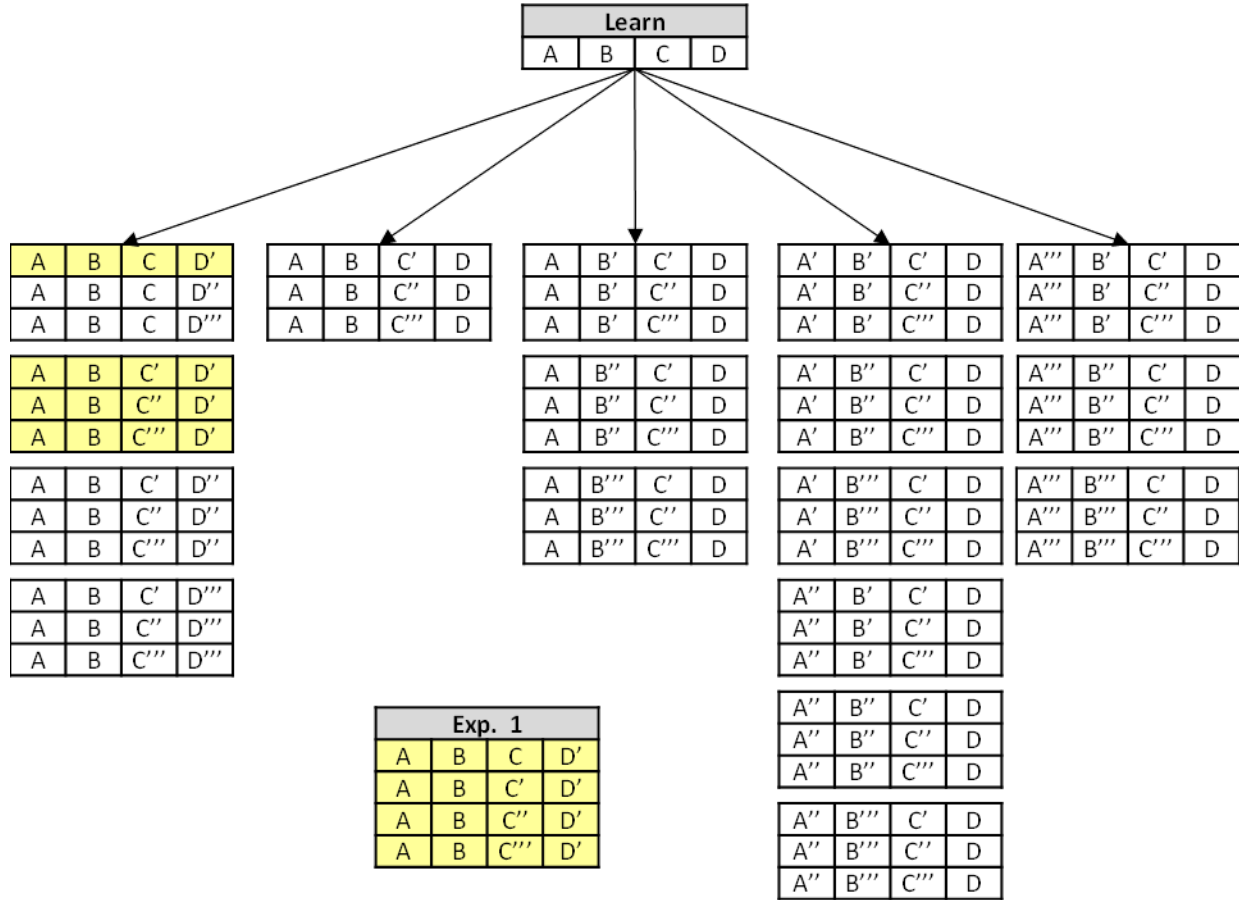


Figure 84: Experimental Protocol For Demonstrating that Spatiotemporal Similarity (G) Reflects the Whole History Leading up to the Current Moment

In the training phase, a single 4-item sequence, [ABCD], is presented once, resulting in a trained network. Then each of four test sequences are presented. In each case, the network is initialized to the trained network state it had immediately after having learned [ABCD]. For each test sequence, we measure similarity of the codes assigned to its 4 moments to the codes assigned to the corresponding moments of the learning trial.

Figure 85 demonstrates the SISC property for Exp. 1. (a) The intersection of the L1 code for the final moment [###D] of each test trial with training moment [ABCD] decreases with the similarity of the 3-item prefix leading up to D and the 3-item prefix, [ABC], that occurred in the training trial. These particular codes resulted from individual runs of each of the test sequences. The spatiotemporal similarity (G_{HV}) of the final sequence moment to the final moment of the learned sequence, [ABCD], falls from panel a to panel d. The size of intersection of the codes of those moments (highlighted in yellow) falls correspondingly. A similar correlation exists for the third sequence moments as well (highlighted in pink). Although mistakes are made on the first and second moments, the number of mistakes remains statistically constant across panels for these moments.

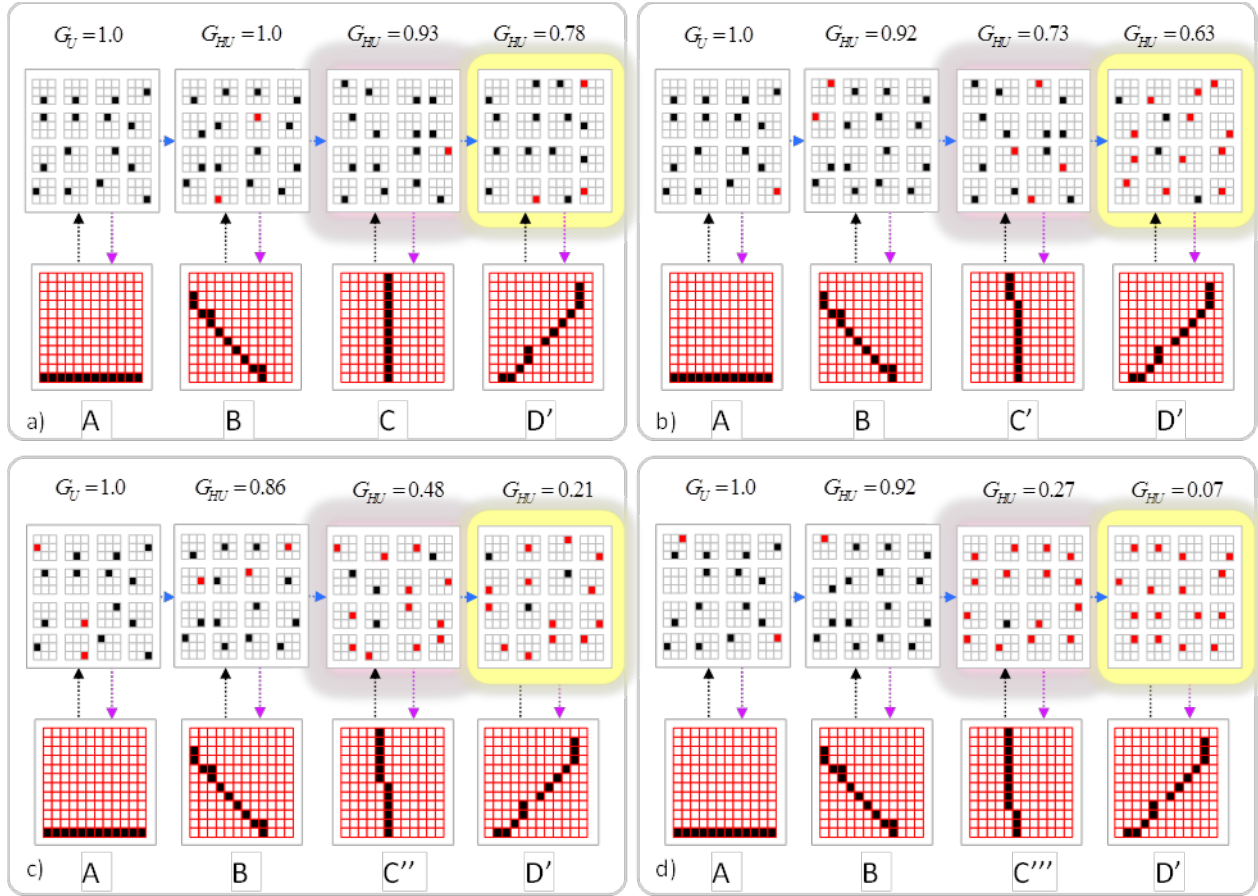


Figure 85: Demonstration of Spatiotemporal SISC Property for 4-Item-Long Sequences

B.5 Conclusions

The studies reported herein constitute a good introduction and overview of what is perhaps the most important property of sparse distributed representations (SDR), the ability to represent similarity in the input space by overlap (intersection) in representation space (code space), “similar-input-similar-codes” (SISC). It is because of this property, that Sparsey simultaneously possesses $O(1)$ time complexity for both storage (learning) and nearest-neighbor retrieval. Possessing simultaneous $O(1)$ time complexity storage and nearest-neighbor retrieval for the case of purely spatial inputs would already constitute a revolution in computing. However, as the studies in the report show, Sparsey achieves this for the case of spatiotemporal inputs, which means that this hitherto unattainable computational efficiency can be applied to all information processing problems involving multivariate time series data, streaming or static DB. This includes real-time vision (including event recognition, understanding) for robotics, surveillance, image/video-based search, speech recognition, language understanding, biosequence analysis (e.g., motif discovery), and any of the myriad “Big Data” applications, e.g., mining huge medical, financial, commercial, manufacturing, economic databases for useful information. At core, the SISC capability stems from three facts:

1. A code is a set of Q units that are chosen independently as draws from probability distributions over separate populations (the CMs).
2. The shape of each of those distributions varies so that the probability of choosing a unit varies directly with the degree of match between the unit's current input and its receptive field (RF).
3. The input fields of all units in all Q CMs must be the same or at least very highly overlapped. Taken together, these facts imply that the expected intersection between two codes varies as a function of the similarity of their corresponding inputs. In particular, if the input fields of all $Q \times K$ units in the coding field (macrocolumn) are not highly overlapped, then the SISC property breaks down. The complete or approximately complete connectivities of all afferent fields to a macrocolumn is required for Sparsey's dynamics to discover/embed the similarity relations (correlations) present in the union of those afferent fields. If one of those afferent fields happens to encode temporal information, i.e., Sparsey's H matrix, then those similarity relations are spatiotemporal in nature.

APPENDIX C - Robust Capability to Handle Complex Sequences

In this study, the training set consists of the four 8-frame sequences shown in Figure 86 (left). This is a *complex* sequence set, meaning that the individual frames occur multiple times in different spatiotemporal contexts. In this case, the input surface, L0, is 12x12 pixels. Figure 86 (right) shows the 3-level network used in this study. L1 consisted of four macrocolumns, each with $Q=16$ CMs, and each CM with $K=9$ cells. Each of the four L1 CMs is fully connected (in both U and D directions) with the 6x6-pixel portion (“aperture”) of the input surface, L0 (indicated for one L1 mac by the transparent blue prism). Given this wiring scheme, each L1 mac will see and assign codes to the spatiotemporal pixel patterns that play out only in its associated aperture. The cells comprising the four L1 macs (there are $16 \times 9 = 144$ cells in each) are completely connected (in both U and D directions) with the 81 cells comprising the L2 mac. Each L1 mac has full recurrent horizontal (H) connectivity with itself and its two immediate neighbors (shown for the upper left L1 mac by the green shading).

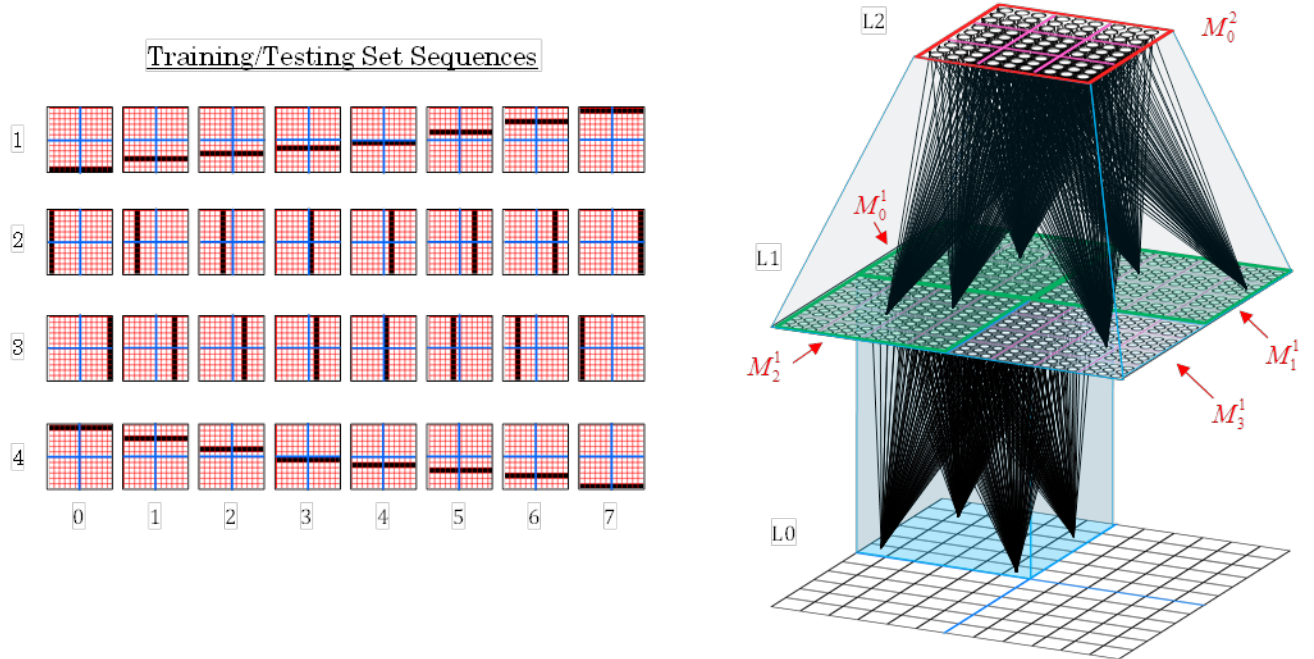


Figure 86: The Four Sequences Comprising the Train/Test Set for the Complex Sequence Study and the Model Used in the Study

During learning, if the number of active pixels in an L1 mac’s aperture \geq a threshold, then that L1 mac will become active and assign a code to the input pattern comprised of those active pixels. Thus, the number of active L1 macs can vary during processing. For example, during the first five frames of Seq. 1, only the bottom two L1 macs, M_2^1 and M_3^1 are active. During the final three frames of Seq. 1, only the top two L1 macs, M_0^1 and M_1^1 are active.

There are many possible policies by which higher-level macs could decide whether or not to become active. The currently implemented policy is that an internal mac, e.g., the single L2 mac, M_0^2 , in this network, becomes active if at least one of the L1 macs in its bottom-up

receptive field (U-RF) is active. In this example, M_0^2 's U-RF includes all four L1 macs. Due to this convergence, codes assigned in M_0^2 can represent spatiotemporal combinations, or compositions, of smaller-scale spatiotemporal pixel patterns playing out in the four L1-scale apertures.

Figure 87 shows the situation in the network when the first frame of Seq. 1 is presented. U signals are sent from the active pixels to M_2^1 and M_3^1 . Only representative subsets of the involved synaptic connections are shown. This is the first input the model experiences and so all weights are still zero. Thus, the U signals arriving at L1 all have zero weight and the u-summations for all cells in M_2^1 and M_3^1 are zero. Nevertheless, codes are activated in M_2^1 and M_3^1 . We denote the code in M_2^1 as $\phi_L(M_2^1, 1, 0)$, meaning the code active in M_2^1 during the learning presentation of frame 0 of sequence 1, i.e., learning *moment* (1,0). These codes are chosen at random via the code selection algorithm (CSA) described in Table 21. Once $\phi_L(M_2^1, 1, 0)$ and $\phi_L(M_3^1, 1, 0)$ are active, they send U-signals to L2. The resulting u-summations for the cells of L2 are also zero and a random code, $\phi_L(M_0^2, 1, 0)$, is chosen in M_0^2 .

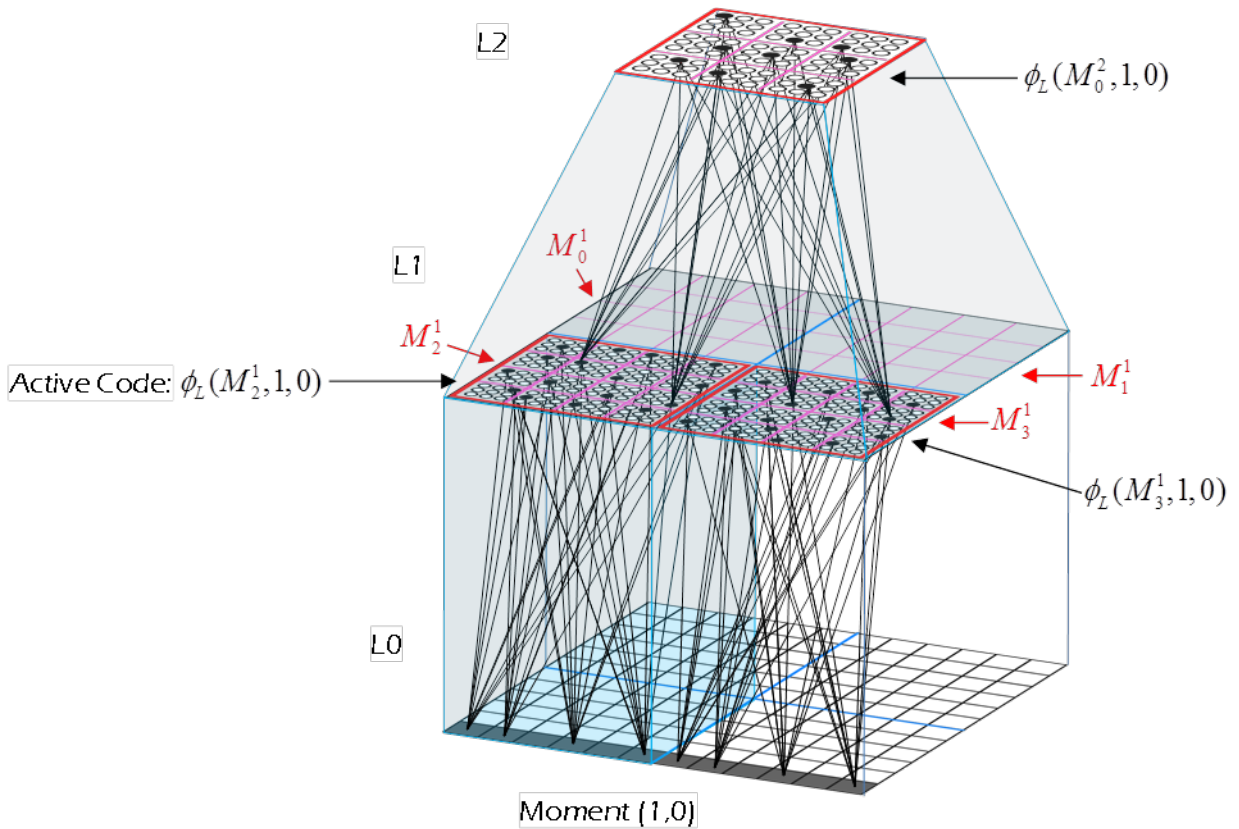


Figure 87: The Situation in the Network when the First Frame of Seq. 1 is Presented

Once the codes are activated at all levels, Hebbian learning occurs. Specifically, the weight from any active cell to any other active cell to which it connects is increased to the maximal value. In our model, weights range from 0 to 127. However, the weights are effectively binary since they are always set to the maximum, 127, when they undergo a Hebbian increase. As discussed in

TR1, weights undergo a gradual passive decay with disuse. However, the overall effect of the learning policy is such that the weights are “effectively binary”.

Figure 88 shows the situation in the network on the second moment, (1,1), of Seq. 1. At L1, we focus on what’s happening in M_2^1 , though the same situation exists in M_3^1 as well. U-signals arise from the new input pattern, but again all of these weights will still be zero. At the same time, H-signals (summarized by green arrows) will be arriving from the codes active on the prior frame, $\phi_L(M_2^1, 1, 0)$ and $\phi_L(M_3^1, 1, 0)$. Winners are shown in M_3^1 to indicate that that is prior code, $\phi_L(M_3^1, 1, 0)$, not the new code that will be chosen in M_3^1 at (1,1). Note that H-signals also arrive recurrently from M_2^1 itself though here we only show the newly chosen code (black cells). Finally, D-signals arrive from the code, $\phi_L(M_0^2, 1, 0)$, that was activated in M_0^2 at moment (1,0) and that remains active at (1,1). Recall that code *persistence* (δ) increases with level; in particular, the persistence of L2 codes, δ_2 is 2. All three sources of input, U, H, and D, to M_2^1 are multiplicatively combined (according to the CSA). Because the h-summations, in particular, are all zero on this frame, the degree of match, G , between the three sources, which is a measure of the spatiotemporal familiarity of the current *moment*, is zero, which ultimately results in choosing the winners in M_2^1 on the current frame (black cells) completely randomly. The same dynamics unfolds in M_3^1 at the same time, also resulting in $G=0$ and a completely randomly chosen code (not shown). As stated above, $\phi_L(M_0^2, 1, 0)$ remains active in M_0^2 on this second frame. Once all macrocolumns at all levels have had their codes updated (modulo persistence), synaptic weights are updated in Hebbian fashion. In particular, on this second frame, the H-weights from the cells comprising $\phi_L(M_2^1, 1, 0)$ and $\phi_L(M_3^1, 1, 0)$ onto the currently active cells in those two macs are increased. And, the H-weights from $\phi_L(M_0^2, 1, 0)$ to $\phi_L(M_0^2, 1, 1)$ are increased: since they are the same code, we can refer to this as an instance of *autoassociation*.

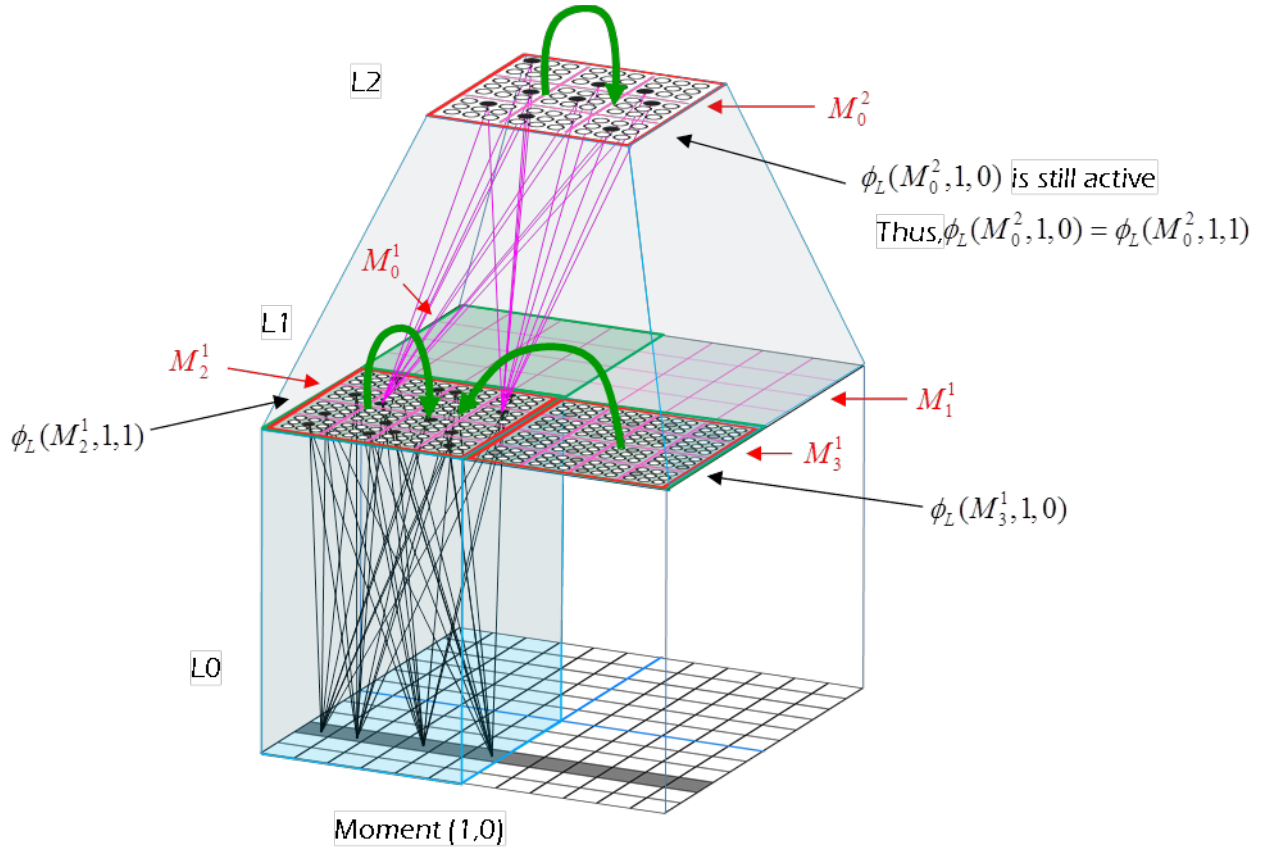


Figure 88: The Situation in the Network on Learning Moment (1,1)

The model proceeds along in this fashion through all eight frames of Seq. 1, with codes being updated on every frame in L1 macrocolumns (since $\delta_1 = 1$) (if sufficiently many pixels are active in the mac's U-RF), codes being updated in every second frame in M_0^2 , and large numbers of U, H, and D weight increases being made within and between macs on the same and neighboring levels. Thus, the processing of a single instance of a sequence leaves behind a strong (because all involved weight increases are maximal), multi-level, spatiotemporal memory trace. Because this strong trace is formed with a single trial, we view this process as a model of human *episodic memory* (Tulving 1983). As our results show (see Figure 92), such traces are sufficient to support highly accurate recognition of subsequent instances of the same sequences. In prior studies, we have also shown that randomly perturbed instances of the training traces can also be recognized with great accuracy. This latter capability—i.e., activating the same exact internal memory trace in response to novel, but similar, instances of complex spatiotemporal input sequence—shows that the model actually builds category representations, which is the basis of *semantic memory* (which is knowledge of the higher-order statistics of the input set) and does so on-the-fly and based on single trials.

Some of our earlier studies involved randomly perturbed instances of training inputs for networks with only a single mac (e.g., see [“Results/Best-Match”](#) tab of Neurithmic Systems homepage). The categories learned in those cases were simple. However, as shown by this study and others reported in the main body of this report, our research has progressed to focusing

on much larger models and larger/longer input sequences, resulting in large and complex patterns of activity involving multiple macs across multiple levels. Our more general goal is to analyze categories defined in higher-level macs whose codes depend on *combinations* of lower-level codes. Our goal is to show the emergence of more complex, nonlinear, categories that could be expressed more in terms of parts-based definitions of categories. Note that as discussed in TR2, we expect that the learning of highly nonlinear categories [as discussed in (Bengio, Courville et al. 2012)] will require the cross-modal “supervised” learning scenario between a-symbolic codes (e.g., those based only on raw visual inputs) and symbolic codes which will be derived from textual labels/annotations that are in register with the a-symbolic input sources (e.g., captions in video).

This complex sequence study has three major points:

1. Showing that the model can recognize *complex sequence* sets—i.e., sequences in which the same state occurs multiple times in different contexts—which is the general condition for language, i.e., the order 10^5 English words are all composed from an alphabet of just 26 letters. Moreover, this can be achieved with single-trial learning.
2. Showing that a sequence’s memory trace, the activation of which constitutes recognition of the sequence, can involve multiple macs operating in tight coordination and where the time-scale of the dynamics differs across levels; here L2’s time-scale is half that of L1.
3. Showing that the model can momentarily deal with *multiple competing hypotheses*. That is, at any given moment, if the total inputs to a mac equally implicate N of the hypotheses (codes) that have been previously stored in the mac, then all N hypotheses will become partially active. Specifically, each hypothesis will (statistically) be represented with Q/N of the cells comprising that hypothesis’ full code. If subsequent inputs are consistent with subsets of the active hypotheses, the inconsistent hypotheses are winnowed out and, with each such winnowing, the degrees of activation of the remaining hypotheses increase until just one is active at full strength (i.e., all Q of that hypothesis’s winners are active).

Figure 89 (left) shows the internal variables that exist during recognition moment (1,0), which is a moment at which two competing hypotheses become active. Note that the trace information (in the charts at bottom) show the variables only for the first six of the $Q=16$ CMs in M_2^1 , but conditions are statistically similar in all 16. The existence of two equally strong hypotheses can be seen in the ρ trace in that in almost every CM, there are two equally probable winners. In each CM, one of these cells is contained in $\phi_L(M_2^1, 1, 0)$ and the other is contained in $\phi_L(M_2^1, 4, 7)$. $\phi_L(M_2^1, 1, 0)$, shown in Figure 87, is repeated here in the inset. The code chosen during learning moment (4,7), $\phi_L(M_2^1, 4, 7)$, is shown in the inset in Figure 89 (right). In this particular case, although the two possible cells are equally probable, it so happens that the winner from $\phi_L(M_2^1, 1, 0)$ is chosen in 75% of the CMs. In the other 25%, the winner is from $\phi_L(M_2^1, 4, 7)$. We denote the code activated on recognition moment (1,0), $\phi_R(M_2^1, 1, 0)$, to distinguish it from the $\phi_L(M_2^1, 1, 0)$. The resolution of the two co-active hypotheses on the next moment (1,1) is shown in Figure 91. But before turning to that figure, we point out that in the other recognition moment where this same input pattern presents, (4,7), shown in Figure 89 (right), only one hypothesis is

present. This is because when recognition moment (4,7) occurs there are H- and D-signals present which, when combined with the U-signals, implicate only code $\phi_L(M_2^1, 4, 7)$.

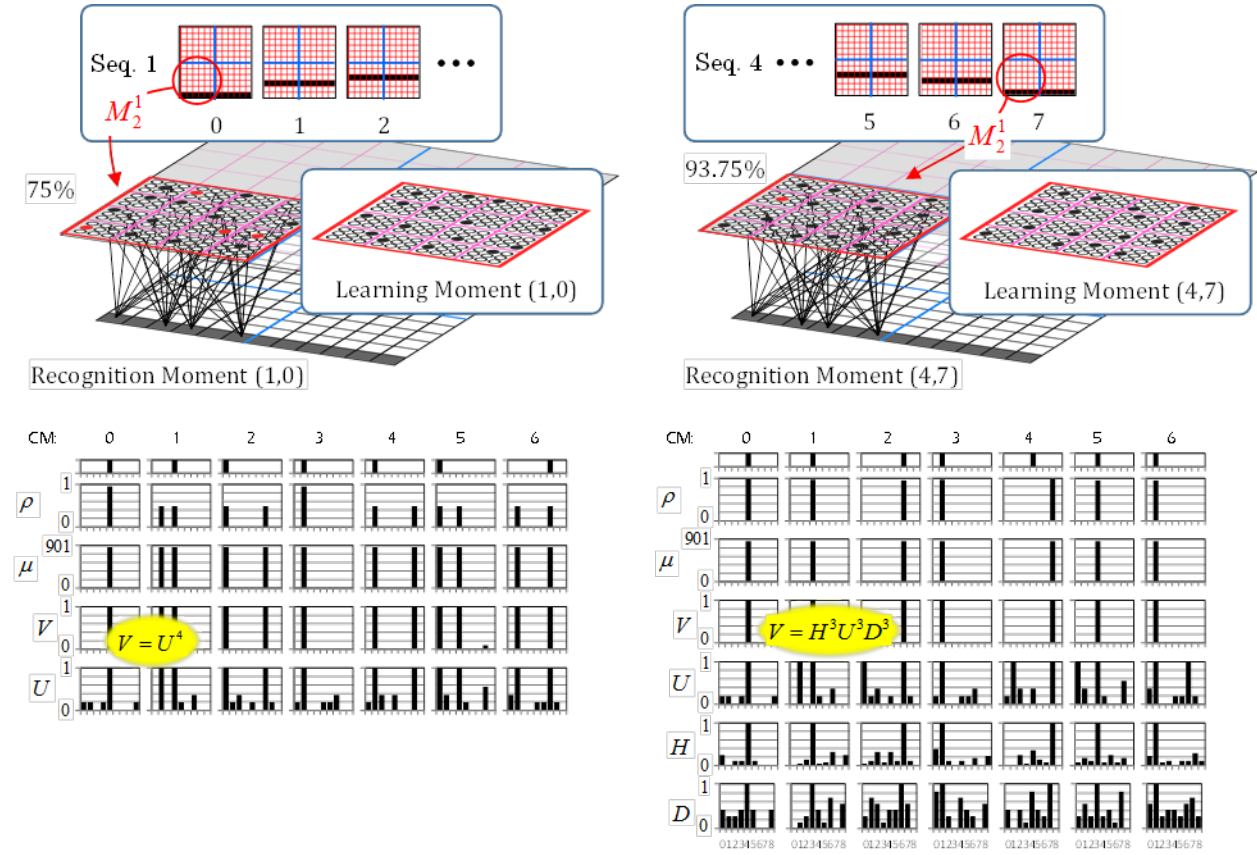


Figure 89: Comparison of Two Recognition Moments, (1,0) and (4,7), in which the Input Pattern is the Horizontal Bar Across the Bottom Row of L0

Figure 90 shows the codes active in both L1 macs and M_0^2 at recognition moment (1,0). It also shows the detailed trace information for M_0^2 , in which the two approximately equally strongly implicated cells can be seen in most of the CMs. Note: the reason why there is only one strongly implicated cell in CM 5 is that the same cell, cell 1, was chosen in both competing codes, $\phi_L(M_0^2, 1, 0)$ and $\phi_L(M_0^2, 4, 7)$. The presence of three approximately equally strongly implicated cells in CMs, 0, 4, and 7, is due to crosstalk between the stored traces.

Note that in M_3^1 and M_0^2 , the numbers of winners from the two competing hypotheses in each mac is much closer to the expected value, $Q/2$. The detailed variable trace information is shown for M_0^2 . The codes for the two possible learning moments that the current moment could be, based only on U-signals, which are the only signals present, are shown at top right. About half the winners chosen at the current moment are from $\phi_L(M_0^2, 1, 0)$ and about half from $\phi_L(M_0^2, 4, 7)$. The winners from $\phi_L(M_0^2, 4, 7)$ are shown in red because they are incorrect, given that the current recognition moment is in fact (1,0), not (4,7).

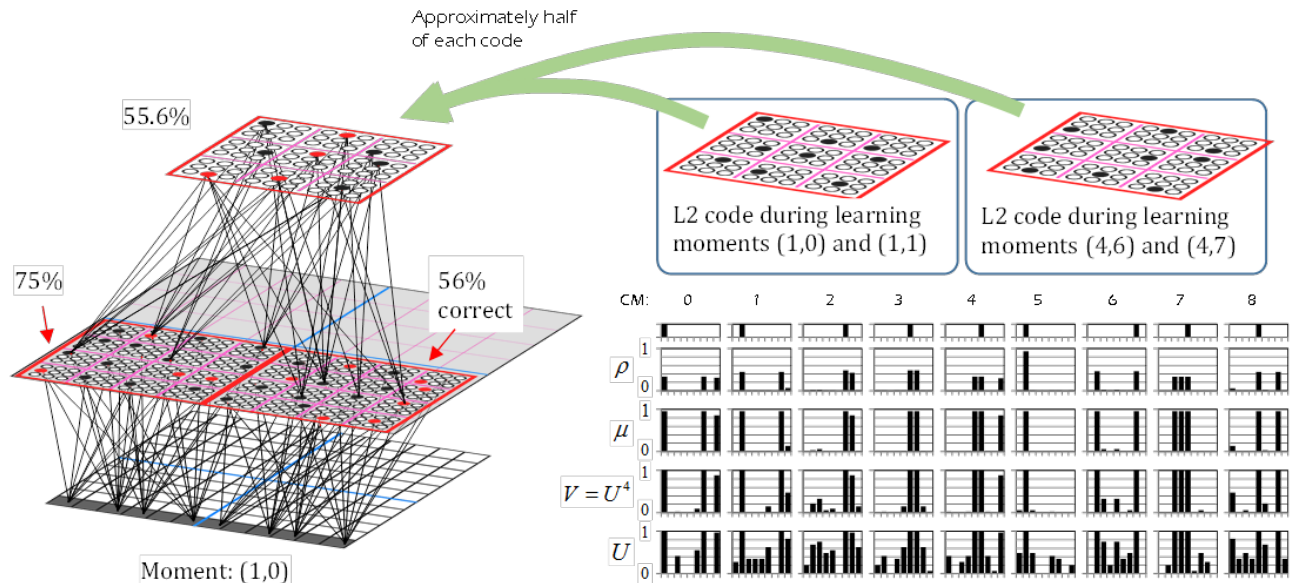


Figure 90: All Three Relevant Codes Activated on Recognition Moment (1,0) and Detailed Trace Information for the L2 Mac Showing Presence of Multiple Competing Hypotheses

Figure 91 shows how the two competing hypotheses in M_2^1 , M_3^1 , and M_0^2 are correctly resolved with the arrival of disambiguating input on the following frame. The conditions at recognition moment (1,0) are repeated at left. Green/magenta arrows suggest, in summary fashion, the H-/D-signals that will influence the choice of codes on the following recognition moment, (1,1). However, note that because the codes at the origins of these arrows all have two hypotheses active, the strength of evidence from both of these sources (H, D) will only be about half as great as they were during the original learning moments. That is, in each of the source macs only about $Q/2$ of the cells contained in either of the two competing hypotheses (codes) is active. Thus, the h- and d-summations at the destination macs will be only about half as great as they were during learning, when the full Q cells representing any of the relevant hypotheses were active.

As described in TR1, the CSA normalizes any of a cell's raw summations, e.g., its h-summation, by dividing by the largest possible h-summation it could have, given the number of active macs in its H-RF. The largest possible h-summation would correspond to the case where all Q cells in each of active macs in its H-RF have (during learning) increased their weights to the cell to their maximum value. In this case, that normalizer would equal $2 \times Q \times (\text{the weight of an increased synapse})$. However, as noted above, in the present instance, the h- and d-summations are only about half as large as they were during learning, which will lead to normalized H and D values of approximately 0.5. This would in turn lead to low V values and ultimately, to low win probabilities for any of the cells in any of the codes of consistent successor moments. However, looking at the trace information for M_2^1 at lower left, the reader can see that the maximum value for both H and D, in all CMs (we show only the data for the first 8 CMs) is 1.0, not 0.5. This is due to the following crucial operational principle of our model. On every frame of every sequence, during both learning and recognition, every macrocolumn computes the number of multiple competing hypotheses that are active. This can be approximated as the average number

of cells per CM that have V close to 1.0. If that average is 2, then the mac knows that there are two approximately equally probable winners in each CM and therefore that, following code selection, there will be two codes active at approximately half strength ($Q/2$ cells active). In this case, the mac simply multiplies its outgoing signals by 2. This means that macs downstream to a given mac, X, will receive full H- and D-support for all hypotheses (codes) that were successors of any of the codes co-active in X at recognition moment (1,0). However, for each of the relevant macs, the new input on recognition moment (1,1) is consistent only with one of those potential successor codes. For example, in M_2^1 , only $\phi_L(M_2^1, 1, 1)$ has the full U, H, and D support to become active. This can be seen in the charts at bottom: only the cells in $\phi_L(M_2^1, 1, 1)$ have U, H, and D all equal to 1 (thin red rectangle shows representative case in one CM). Thus, only one cell has very high probability of winning in each CM, resulting in a much higher accuracy on recognition moment, (1,1). Similar considerations apply to M_3^1 and M_0^2 as well.

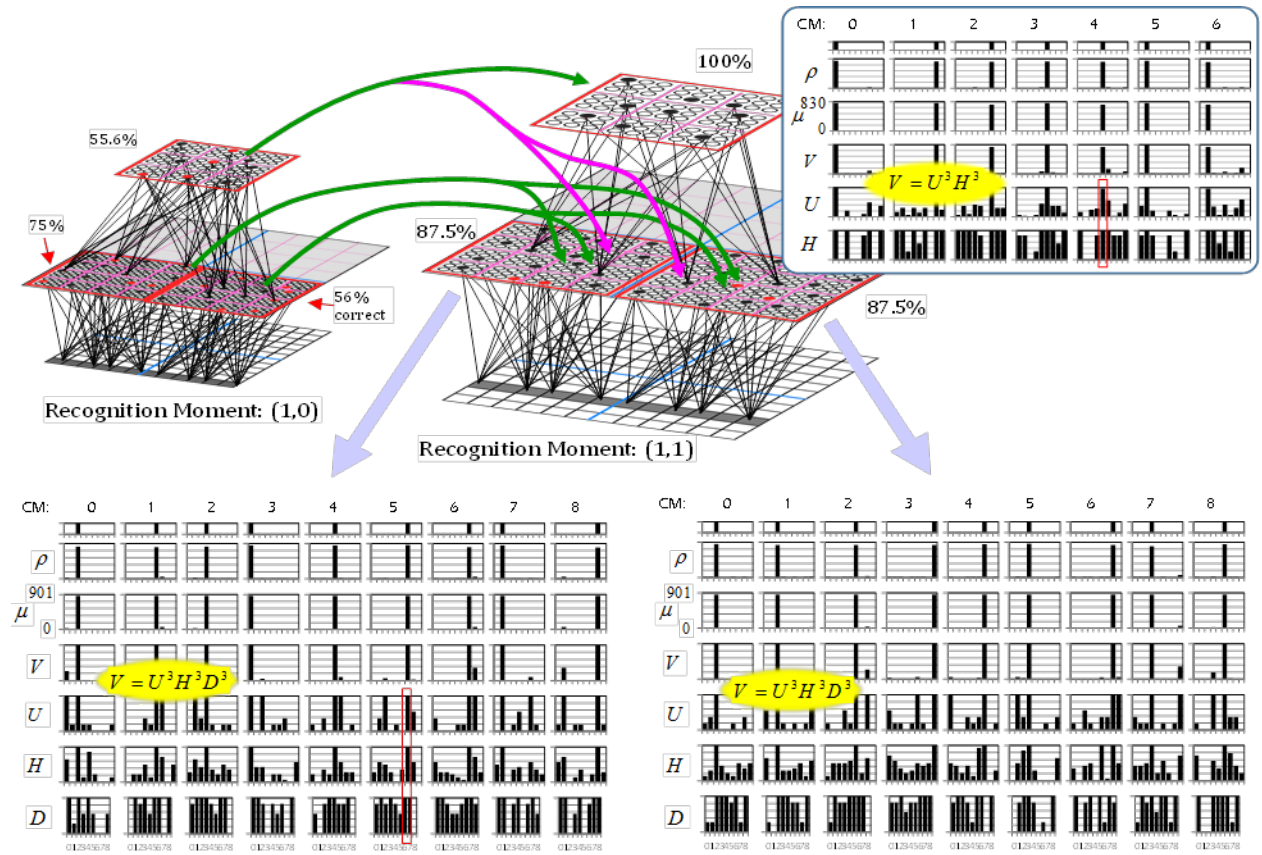


Figure 91: Correct Resolution of Multiple Competing Hypotheses based on Temporal Context Signals Mediated by H and D Inputs

(ep: 0) Level 2:	R_star:	0.88:	R_omega:	0.89
(ep: 0) Level 1:	R_star:	0.89:	R_omega:	0.97
(ep: 1) Level 2:	R_star:	0.94:	R_omega:	0.89
(ep: 1) Level 1:	R_star:	0.91:	R_omega:	1.00
(ep: 2) Level 2:	R_star:	0.88:	R_omega:	1.00
(ep: 2) Level 1:	R_star:	0.95:	R_omega:	0.97
(ep: 3) Level 2:	R_star:	0.93:	R_omega:	0.78
(ep: 3) Level 1:	R_star:	0.93:	R_omega:	0.97
RUN 0:	R_star (all levs):	0.91	R_omega (all levs):	0.93
AVE OVER ALL RUNS:	R_star (all levs):	0.91	R_omega (all levs):	0.93

Average accuracy on last frame of sequence, over all sequences, over all active macs at all levels.

Full Recognition Trace of Sequence 1

L2: [0, 0]: G_U 1.000	RA:	55.56%	0 1 6 5 5 1 7 4 4
L1: [0, 0]: (active macs: 2/ 4)	RA:	75.00%	4 3 0 1 1 0 6 5 0 3 5 4 3 3 3 1
MAC 2: G_U 1.000	RA:	56.25%	7 4 4 6 8 7 5 1 4 7 1 1 3 2 7 4
MAC 3: G_U 1.000	RA:		132 133 134 135 136 137 138 139 140 141 142 143 ...12 active f
L0: [0, 0]:	RA:		
L2: [0, 1]: G_HU 0.964	RA:	100.00%	0 7 6 5 4 1 1 3 4
L1: [0, 1]: (active macs: 2/ 4)	RA:	87.50%	2 5 3 0 4 6 6 1 7 8 1 6 4 8 7 4
MAC 2: G_HU 1.000	RA:	87.50%	2 2 5 8 6 3 7 3 5 2 8 1 3 8 1 0
MAC 3: G_HU 1.000	RA:		108 109 110 111 112 113 114 115 116 117 118 119 ...12 active f
L0: [0, 1]:	RA:		
L2: [0, 2]: G_HU 1.000	RA:	88.89%	8 5 1 3 5 1 4 2 0
L1: [0, 2]: (active macs: 2/ 4)	RA:	100.00%	4 1 7 0 3 1 0 0 4 7 7 6 7 3 2 8
MAC 2: G_HU 0.971	RA:	93.75%	4 3 6 7 2 2 5 6 8 7 6 3 8 1 3 0
MAC 3: G_HU 0.965	RA:		96 97 98 99 100 101 102 103 104 105 106 107 ...12 active f
L0: [0, 2]:	RA:		
L2: [0, 3]: G_HU 0.955	RA:	100.00%	8 5 1 5 5 1 4 2 0
L1: [0, 3]: (active macs: 2/ 4)	RA:	100.00%	2 8 0 8 4 1 0 7 7 3 7 5 5 1 6 8
MAC 2: G_HU 1.000	RA:	100.00%	3 2 2 4 1 8 6 2 7 6 8 3 2 2 6 6
MAC 3: G_HU 1.000	RA:		84 85 86 87 88 89 90 91 92 93 94 95 ...12 active features
L0: [0, 3]:	RA:		
L2: [0, 4]: G_HU 1.000	RA:	100.00%	0 0 8 4 8 0 4 6 3
L1: [0, 4]: (active macs: 2/ 4)	RA:	81.25%	0 6 2 8 6 0 2 1 0 6 8 6 7 3 1 7
MAC 2: G_HU 1.000	RA:	100.00%	8 5 2 7 5 3 3 3 6 2 1 0 8 4 6 5
MAC 3: G_HU 1.000	RA:		72 73 74 75 76 77 78 79 80 81 82 83 ...12 active features
L0: [0, 4]:	RA:		
L2: [0, 5]: G_HU 1.000	RA:	66.67%	0 1 8 7 8 0 4 7 3
L1: [0, 5]: (active macs: 2/ 4)	RA:	93.75%	4 1 5 5 3 1 6 5 3 6 1 1 6 1 2 3
MAC 2: G_HU 0.798	RA:	87.50%	6 2 5 6 5 5 5 8 1 1 0 5 0 1 1 6
MAC 3: G_HU 1.000	RA:		48 49 50 51 52 53 54 55 56 57 58 59 ...12 active features
L0: [0, 5]:	RA:		
L2: [0, 6]: G_HU 0.879	RA:	100.00%	2 5 0 6 5 8 6 1 8
L1: [0, 6]: (active macs: 2/ 4)	RA:	81.25%	7 3 6 6 2 7 7 8 7 7 2 6 2 3 2 7
MAC 2: G_HU 0.833	RA:	93.75%	6 4 0 5 3 5 4 8 1 2 4 5 6 8 5 5
MAC 3: G_HU 0.853	RA:		24 25 26 27 28 29 30 31 32 33 34 35 ...12 active features
L0: [0, 6]:	RA:		
L2: [0, 7]: G_HU 1.000	RA:	88.89%	2 5 0 6 5 8 6 3 8
L1: [0, 7]: (active macs: 2/ 4)	RA:	100.00%	2 2 6 1 7 5 5 2 2 4 6 1 2 1 8 6
MAC 2: G_HU 0.971	RA:	93.75%	2 6 3 0 6 0 6 5 6 0 5 5 3 3 2 4
MAC 3: G_HU 0.977	RA:		0 1 2 3 4 5 6 7 8 9 10 11 ...12 active features
L0: [0, 7]:	RA:		

Figure 92: The Overall Recognition Performance for All Four Sequences

The overall recognition performance for all four sequences is shown in at the top of Figure 92. The detailed frame-by-frame, and mac-specific codes and accuracies are given only for Seq. 1. However, the performance was similar for the other three sequences as well. There are many points of discussion possible regarding this data. At present, we simply emphasize that:

1. The complex, space-time, multilevel traces of the 4 sequence, containing many duplicate instances of input states in different contexts, which are constructed on-the-fly based on single trials, reactivate with nearly perfect fidelity, evincing strong recognition capability.
2. The model moves transiently through ambiguous moments, resolving the ambiguity based on incremental presentation of disambiguating evidence.
3. Individual CM-level mistakes occur with some statistical regularity (which depends on many model parameters). However, these mistakes are continually corrected on each frame.

APPENDIX D - Mechanisms Underlying Invariant Recognition

Figure 93 is an expanded version of Figure 25. It shows the U-summations and V values of all cells in the relevant macs during the recognition trial of the original training frame (“test=train”, top of figure) and the noisy test frame (“test≠train”, bottom). Figure 93a shows the overall 64x64 frame; the inset zooms in on the particular patch of the pixels (cyan) that is the input level (L0) receptive field (RF) of M_{402}^2 , which we denote $F_U(2,402,0)$. But, L2 does not receive U-inputs directly from L0, but rather via L1. The yellow inset shows the set of L1 macs (cyan and purple) comprising M_{402}^2 ’s L1 RF, denoted $F_U(2,402,1)$. There are 10 L1 macs in the set; two are active (purple), M_{773}^1 and M_{854}^1 . The same set is seen in perspective in the yellow ellipse, but the blue lines, representing a tiny sample of the U-wts comprising the relevant RFs, make them hard to see. Figure 93b is present to make clear the relation of the three levels in this example (though the network in this case has nine levels, as can be seen in the inset of Figure 25a).

Figure 93d shows the relevant signals present in M_{773}^1 while processing this frame. The seven bar plots in each row correspond to the $Q=7$ CMs comprising M_{773}^1 ; the x-axis of each plot ranges over the $K=7$ cells comprising each CM. The bottom row of plots shows the raw U summations (u), the next one, the normalized U summations (U), then U^2 , and then the V values, which in this case, since only U signals are present, equal the U^2 values. The green-bordered inset zoom in on the L0 RF of M_{773}^1 , $F_U(1,773,0)$, which has five active pixels (black) out of 19. $F_U(1,773,0)$ is also shown superimposed on $F_U(2,402,0)$ in the inset above. Again, the input frame here was presented exactly on a prior learning trial. Due to the learning that would have taken place on that learning trial, when it presents again here, the U-signals from the active pixels to M_{773}^1 will yield high U-summations specifically for the cells that won on that learning trial.¹⁴ The single black bars in the plots correspond to those winning cells, i.e., the sparse distributed code (SDC), or *cell assembly* (CA), chosen to represent the input to M_{773}^1 at that time. The fact that all 7 winners have $V=1$ here, yields $G=1.0$, which means that this input is perfectly familiar to this mac. In the experiments reported herein, we used the max-V CSA version to pick winners during recognition, so the CA activated in M_{773}^1 on the learning trial is perfectly reinstated, i.e. M_{773}^1 has recognition accuracy, $R=100\%$.

Figure 93e shows similar information for M_{402}^2 ’s other active afferent L1 mac on this occasion, M_{854}^1 . The inset shows M_{854}^1 ’s RF, $F_U(1,854,0)$, which has five out of 17 active pixels. $F_U(1,854,0)$ is also shown superimposed on $F_U(2,402,0)$ in the inset in Figure 11a, which shows how the L1 mac RFs overlap. The situation in M_{854}^1 is qualitatively the same as for M_{773}^1 : the input is the same as during learning and the same CA is activated ($R=100\%$).

¹⁴ In general, other cells will also have non-zero U-summations (due to crosstalk), but in this example, since only three 8-frame snippets were presented during learning, there is no crosstalk.

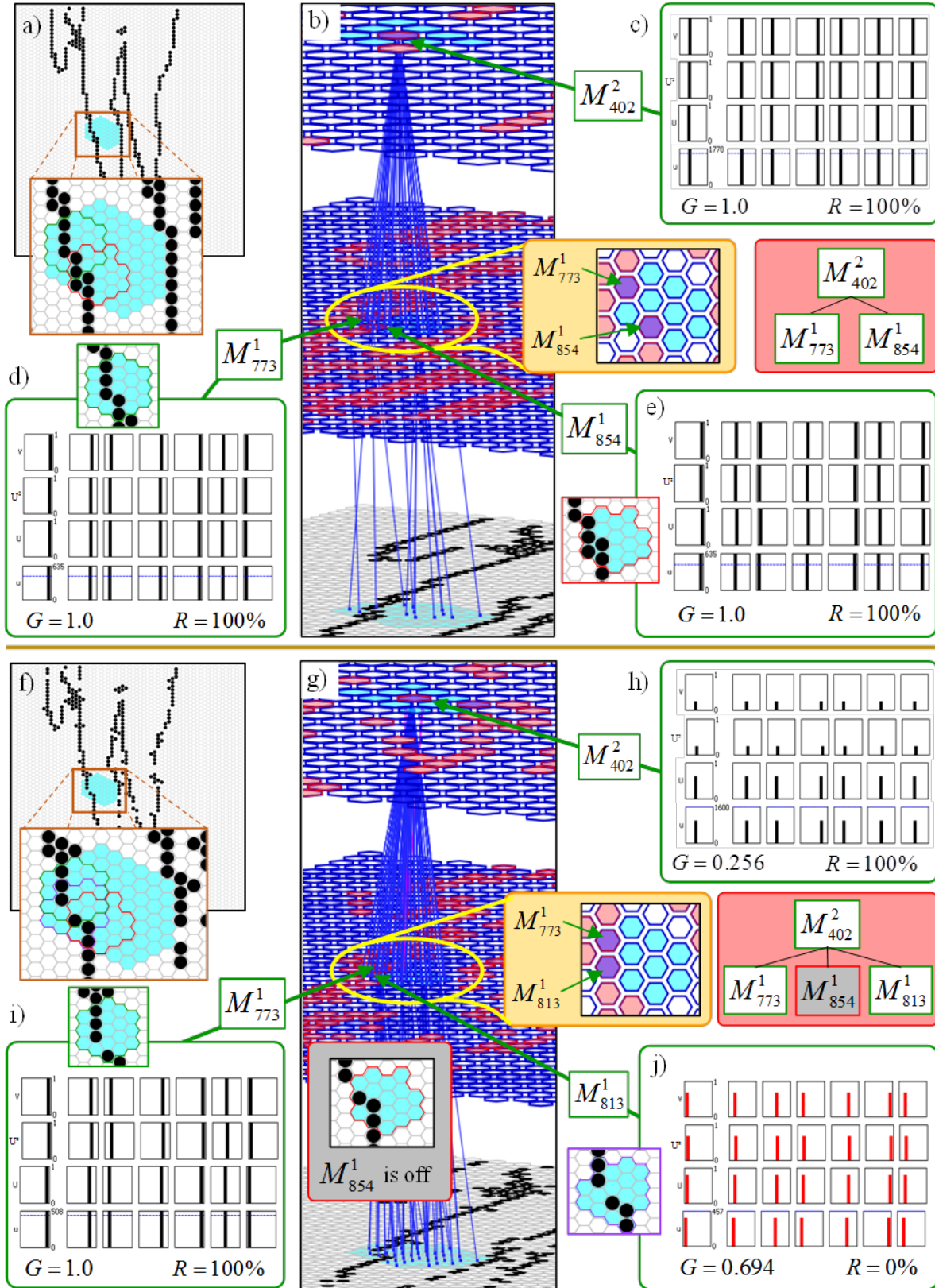


Figure 93: Demonstration of Invariant Recognition

Once the two L1 macs, M_{773}^1 and M_{854}^1 , are activated, they send U-signals to M_{402}^2 . Figure 93c shows the resulting signals in M_{402}^2 . In this case, each of the cells comprising the CA that won in M_{402}^2 on the learning trial will be receiving a u-input from 14 cells, the 7 winners in M_{773}^1 and the 7 winners in M_{854}^1 . Since these weights will all have been maximized, i.e., set to 127, on the learning trial, their U-summations will be $14 \times 127 = 1,778$. As in the case for the two L1 macs, the CA activated in M_{402}^2 is identical to that activated on the learning trial, $R = 100\%$.

Now we describe the situation when processing the noisy version of the input. Figure 93f shows the noisy 64x64 input and the inset shows M_{402}^2 's RF with the RFs of the three L1 macs, M_{773}^1 , M_{854}^1 , and M_{813}^1 , superimposed. There are several points explaining M_{402}^2 's invariant response:

1. Compare Figure 93i to Figure 93d. The only difference is that in Figure 93i, there are only four active pixels, whereas in Figure 93d, there were 5. This leads to lower U-summations for the winning cells, 508 vs. 635. However, the threshold is set at 508 in both cases, which yields $U=1.0$ for every winning cell, and ultimately $G=1.0$. Since the max-V CSA is used, the exactly correct code is reinstated ($R=100\%$). This illustrates one invariance mechanism at play in this example.
2. As explained in Figure 25, M_{854}^1 is not active because it has only three active pixels. Thus, it does not send any U-signals to M_{402}^2 , indicated by M_{854}^1 being grayed out. This wholesale omission of an entire mac's worth of U-signals relative to the original training input seems like it should present quite a problem, but it does not as we shall see.
3. L1 mac M_{813}^1 , which is not active for the original training input, is active for noisy input. Again, such a large-scale perturbation to the inputs arriving at M_{402}^2 would seem problematic. However, because M_{813}^1 was not active on the learning trial for this input, none of the cells currently active in M_{813}^1 have increased their weights onto any of the cells in M_{402}^2 . In general, if we had stored many more snippets in this model, there would be some crosstalk coming from the cells active in M_{813}^1 , but in this example there is none. So although there is an intruding feature present in the input, it exerts no effect on the code selection process in M_{402}^2 . This is a 2nd invariance mechanism at play in the example.

Finally, if we now look at the signals present in M_{402}^2 (in Figure 93h) we see a third principle of invariance. Even though the u-summations to the cells that won for the training input are lower than they were for the training input—in fact, they are only about half as high as they were for the training input (because of the absence of the signals from M_{854}^1)—those cells still have the max V 's in their respective CMs. Thus, those cells win. Note that even though the G value in M_{402}^2 is only 0.256, the entire code chosen for the original input is exactly reinstated for the noisy input ($R = 100\%$). Thus, as noted earlier, this use of the max-V CSA constitutes a third, very powerful, mechanism of invariance in the model.

Figure 94 shows are more summary view of Figure 25 in which, despite very different input pattern from L1, the same code is activated in M_{402}^2 (actual code not shown). The large difference in M_{402}^2 's immediate inputs, the difference between X and X' , at the input level (L0),

was rather small; compare the insets in the lower right of each panel. Because, in this example, X and X' are actually quite similar, one might conclude that the ultimate invariance shown at L2 (by M_{402}^2) is not that impressive. However, in our experiments, we are finding the size of the differences tolerated, i.e., the degree of invariance, increases with level. We provide two detailed examples of this, one for L4 mac M_{152}^4 (Figure 96); one for an L6 mac M_{19}^6 (Figure 97).

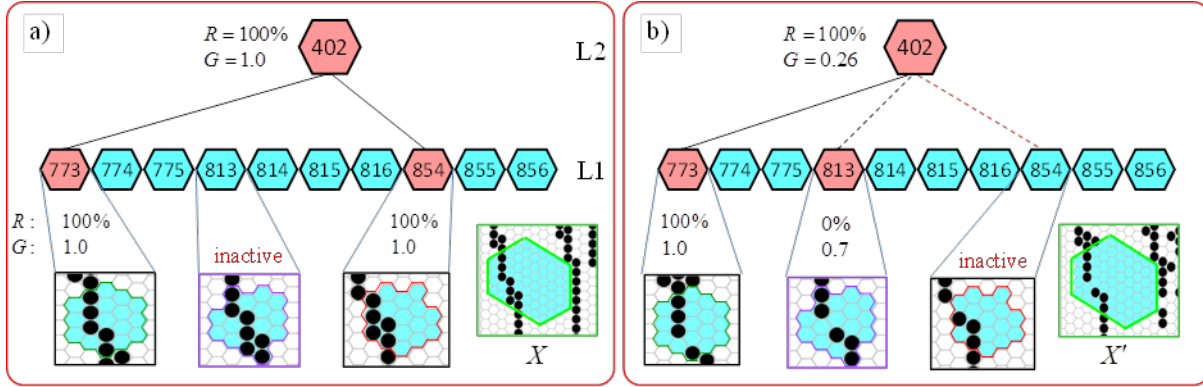


Figure 94: Invariance: The Same Code Activates in M_{402}^2 Despite Very Different Immediate Input Patterns from L1

Figure 95 summarizes those two instances and provides six more from macs at L5 (panels c-h), providing a small sample of the large and complex invariances exhibited by macs in our simulations. In each panel, we show the portion of the overall input present in the L0 receptive field (cyan or outlined in green) of a given mac for a repeat of a learned input (X) and a presentation of a noisy version of that learned input (X'). The nomenclature $X(e2, f7, M_{19}^6)$ is read as “the subset of the active pixels in M_{19}^6 ’s L0 RF when frame 7 of learned episode e2 is presented, which actually end up being coded by the sparse distributed code active in M_{19}^6 ”.

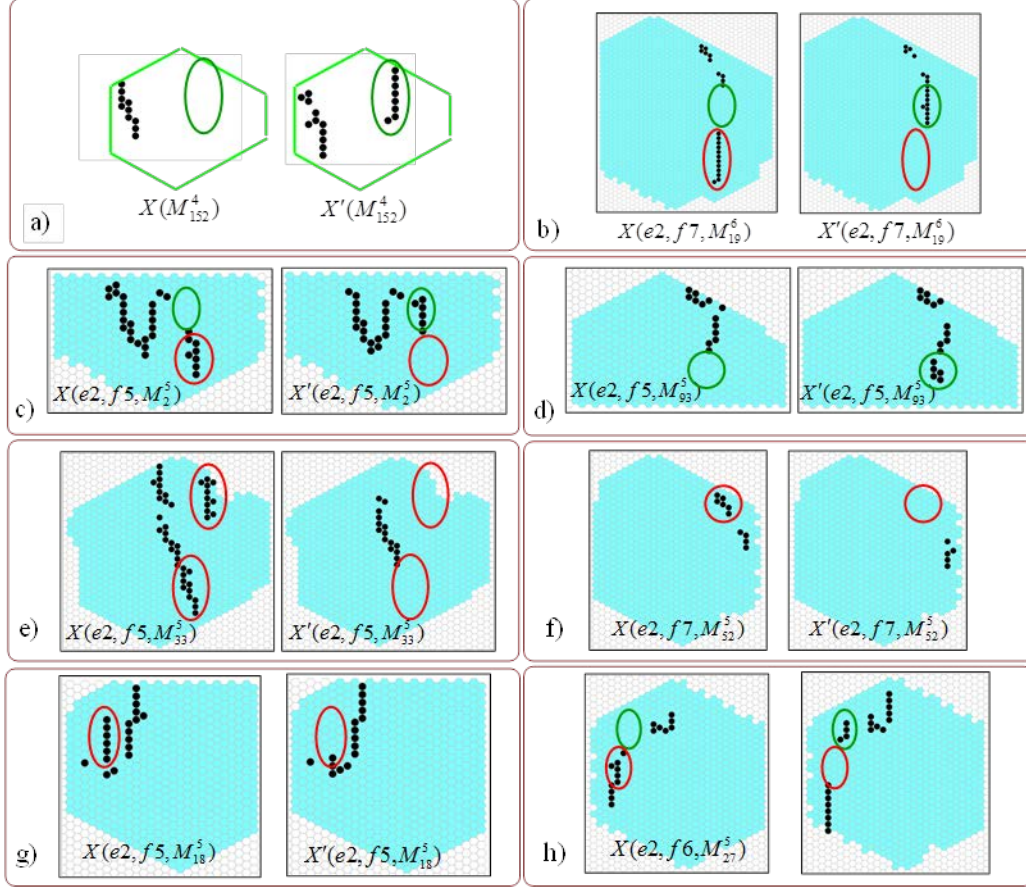


Figure 95: Examples of Invariances Learned

It is crucial to understand that in all these cases, the exact same code is activated in the relevant mac for X and X' . That is, in each case, the mac is *equating* these two inputs: it is *recognizing* X' as *identical* to X . Thus, in Figure 95b, when $X(e2, f7, M_{19}^6)$ occurred during learning, the code $\phi(X(e2, f7, M_{19}^6))$ was selected (activated) and, by virtue of the learning that took place, stored in M_{19}^6 , as one of its basis vectors (dictionary elements).¹⁵ The fact that $\phi(X(e2, f7, M_{19}^6))$ is activated in response to $X'(e2, f7, M_{19}^6)$ means that M_{19}^6 considers $X(e2, f7, M_{19}^6)$ to be its closest matching basis element to $X'(e2, f7, M_{19}^6)$ even though $X(e2, f7, M_{19}^6)$ has one part that $X'(e2, f7, M_{19}^6)$ does not (red ellipse) and $X'(e2, f7, M_{19}^6)$ has one part that $X(e2, f7, M_{19}^6)$ lacks (green ellipse). To evaluate how reasonable this is, we would need to move to larger examples in which macs have stored many more elements (codes) and compute some kind of average measure of how appropriate / reasonable the invariances are, across substantial numbers of (X, X') pairs. We will pursue this type of analysis in future work.

¹⁵ We have several variations of our notation for codes. Most generally, we use to denote a code, i.e., a sparse distributed code. Sometimes, use sub/super-script to denote the mac in which the code occurs, the frame on which it occurred, or other information. Here, for simplicity, $\phi(X(M_{19}^6))$, denotes the code in M_{19}^6 that activates in response to the input, $X(M_{19}^6)$.

In conjunction with Figure 25 of the main body and the rest of the figures in this appendix, Figure 96 and Figure 97 show that the extent and nature of the differences between X and X' that are tolerated (responded to invariantly) increases and becomes more complex at progressively higher levels.

Figure 96 demonstrates invariant responding at L4 mac M_{152}^4 . Figure 96a shows which macs are active within M_{152}^4 's U-RF at all lower levels. M_{152}^4 's immediate U-RF consists of the seven L3 macs shown, two of which, M_{274}^3 and M_{296}^3 , are active. Though not shown here, $G=1.0$ and $R=100\%$ for both of these L3 macs in this case. The connecting lines to a level J mac show the immediate U-RFs from level J-1. Black lines are from active macs (purple) and gray, from inactive (cyan at L2 and L3, white at L1). Thus, we can see that M_{274}^3 's U-RF consists of seven L2 macs of which M_{432}^2 and M_{460}^2 are active, that M_{296}^3 's U-RF also consists of seven L2 macs in which M_{460}^2 and M_{490}^2 are active, and that these two U-RFs overlap (they share four of seven L2 macs). Note that M_{152}^4 's U-RF at L2 consists of the 21 L2 macs shown. Continuing down the hierarchy, we see that M_{432}^2 's U-RF consists of eight L1 macs of which M_{856}^1 and M_{897}^1 are active, etc. M_{152}^4 's U-RF at L1 consists of the 68 L1 macs shown. At bottom right of Figure 96a-i, we show actual input pattern in M_{152}^4 's input-level U-RF (outlined in green). This U-RF is the union of the U-RFs of the 68 L1 macs comprising M_{152}^4 's L1 U-RF; we show the U-RFs for three of these L1 macs, M_{856}^1 , M_{897}^1 , and M_{937}^1 . Not all active pixels falling within a given higher-level mac's L0 U-RF actually end up being coded by that mac due to the multiple stages of nonlinear thresholding occurring in intervening levels. Figure 96a-ii shows the subset of the active pixels that actually end up being coded by the sparse distributed code active in M_{152}^4 , seen at upper left. We'll denote this subset as $X(M_{152}^4)$ (read as "X as seen by M_{152}^4 ").

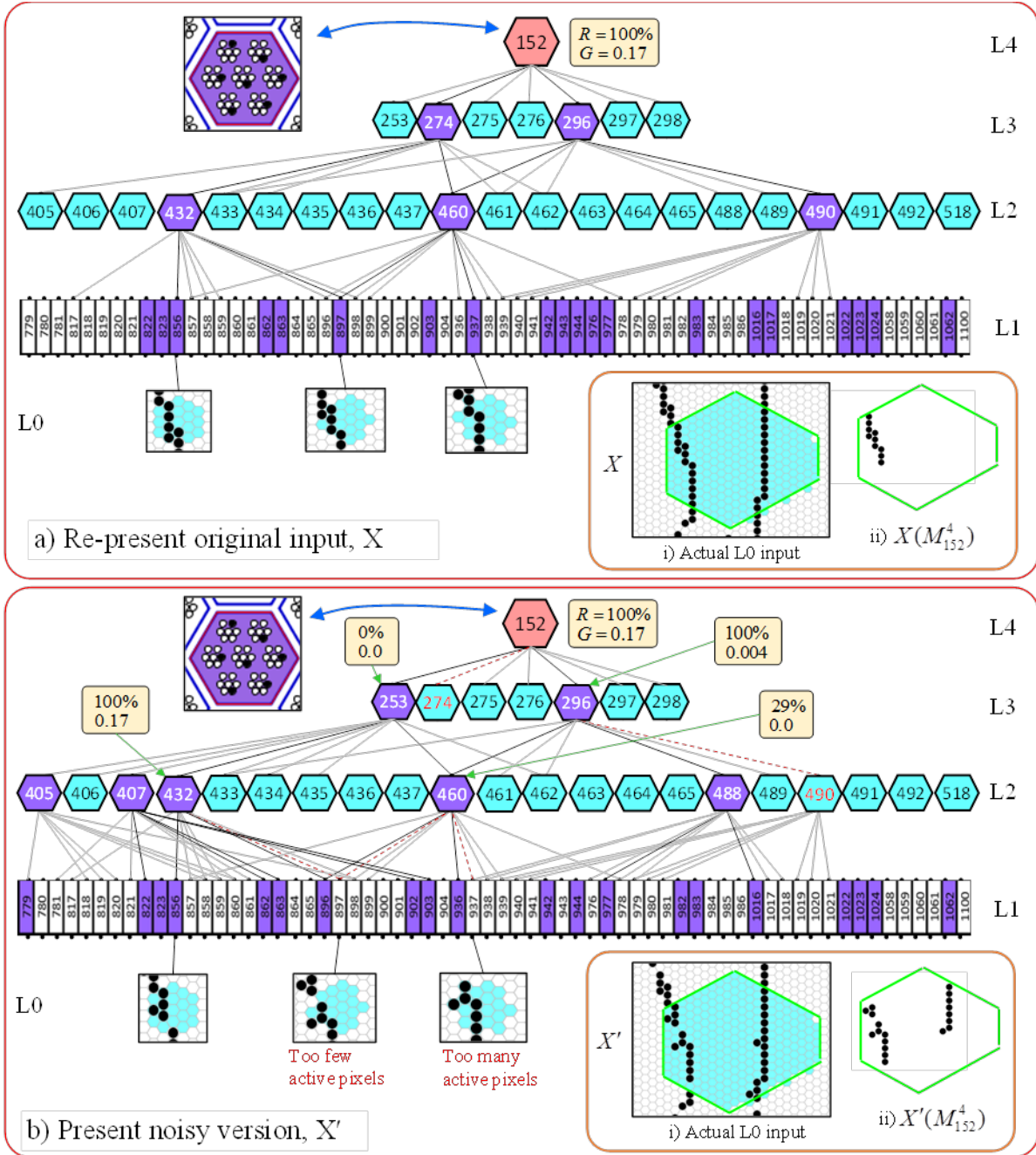


Figure 96: Invariant Recognition at L4 Mac M^4_{152}

Figure 96b shows the situation when a noisy version of the input is presented; compare the “Actual L0 input” figures at lower right of each panel. We make the following points regarding the figure.

1. The same exact code is active in M^4_{152} for the noisy input at well. i.e., M^4_{152} has equated the two actual input patterns, X and X' .

2. There is a pattern across levels in the differences between the codes for X and X' . Specifically, X and X' are actually quite similar at L0. The two L1 codes also have high overlap, i.e., the set of active (purple) macs at L1 for X has high overlap with that for X' . The codes then diverge more at levels L2 and L3. Finally, as we already pointed out, the L4 codes are identical. Thus, there seems to be increasing separation of codes through some levels, which at some point, reverses to become increasing completion.
3. Although the two L0 patterns within M_{152}^4 's L0 U-RF are quite similar, the “as seen by” subsets, $X(M_{152}^4)$ and $X'(M_{152}^4)$, are quite different.
4. There exist multiple paths up through the levels by which a pixel or an active mac at any level can influence a higher level mac. E.g., the pixels causing L1 mac M_{897}^1 to activate in Figure 96a influence code selection in M_{432}^2 and M_{460}^2 , which both influence code selection in M_{274}^3 . Thus, due to the overlapping U-RFs, any particular feature (at L0, a pixel is a feature; at any higher level, the code active in a mac represents a feature) generally has multiple “chances” to influence (be represented by) codes at higher levels.
5. In Figure 96a, note that L2 mac M_{490}^2 is active even though none of its 10 afferent L1 macs is active. What’s happening here is that M_{490}^2 was activated on the prior frame and it has a persistence of two frames. So it remains active on this frame even though its U inputs are no longer such that they would cause it to activate if it was not already active.

We complete our explanation of how progressively higher macs can represent progressively larger-scale invariances with an example of an L6 mac, M_{19}^6 , in Figure 97. We do not show the degree of detail in this figure as we did in the last because it would get too dense. At upper left of each panel, we see the whole 9-level network, in which M_{19}^6 is highlighted in purple at L6. At upper right of each panel, we show that the code activated in both instances is identical. At middle, we show the L0 U-RF of M_{19}^6 (cyan), which includes a large region of L0: the actual input is superimposed. Note there are 36 L6 macs, so their L0 U-RFs are highly overlapped. At lower left of each panel, we show the subset of pixels that end up “surviving” the effects of the multiple intervening levels of nonlinear thresholding and influencing the code selection process in at least one L6 mac. Using our “as seen by” naming convention, we refer to these subsets as $X(L6)$ and $X'(L6)$, respectively. In general, we’d like the fraction of surviving pixels to remain quite high at all levels, perhaps 90% or so, so we’re a little low in the case shown here. Finally, at lower right of each panel, we show the subset of pixels *actually seen by* (and thus, actually influencing code selection in) M_{19}^6 , $X(M_{19}^6)$ and $X'(M_{19}^6)$, respectively. Thus, again we see a substantial degree of invariance (tolerance) in M_{19}^6 ’s responding. These two inputs could both be characterized as a “vertical-ish contour with slight bend at top”, but there are significant differences, e.g., $X'(M_{19}^6)$ has a large gap where $X(M_{19}^6)$ actually has a contour and the centroid of $X(M_{19}^6)$ is noticeably higher (~ 10 pixels) than that of $X'(M_{19}^6)$.

It is crucial to understand that since the exact same code is activated in M_{19}^6 in both cases, it is *equating* these two inputs. It is *recognizing* $X'(M_{19}^6)$ as *identical* to $X(M_{19}^6)$. That is, when $X(M_{19}^6)$ occurred during learning, the code $\phi(X(M_{19}^6))$ was selected (activated) and, by virtue of the learning that took place, stored in M_{19}^6 , as one of its basis vectors (dictionary elements).¹⁶ The fact that $\phi(X(M_{19}^6))$ is activated in response to $X'(M_{19}^6)$ means that M_{19}^6 considers $X(M_{19}^6)$ to be its closest matching basis element to $X'(M_{19}^6)$. To evaluate how reasonable this is, we need to move to larger examples in which macs have stored many more elements and compute some kind of average measure of how appropriate / reasonable the invariances are, across substantial numbers of (X,X') pairs.

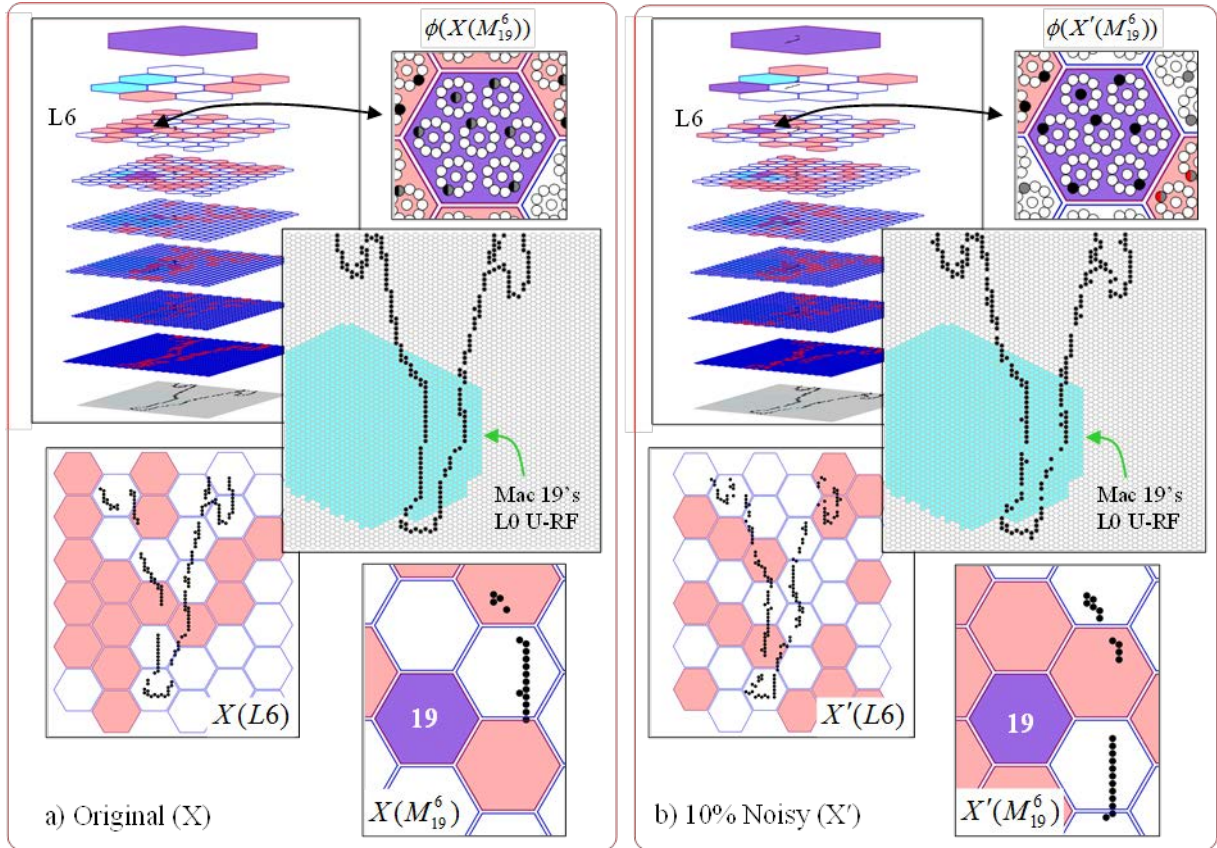


Figure 97: Invariant Recognition at L6 Mac M_{19}^6

¹⁶ We have several variations of our notation for codes. Most generally, we use to denote a code, i.e., a sparse distributed code. Sometimes, use sub/super-script to denote the mac in which the code occurs, the frame on which it occurred, or other information. Here, for simplicity, $\phi(X(M_{19}^6))$, denotes the code in M_{19}^6 that activates in response to the input, $X(M_{19}^6)$.

Figure 98 shows the portions of the input, in other words, the *features* or *parts*, coded by each of the 16 active L6 macs on frame 7 of input snippet. We note the following:

1. There is redundancy: certain subsets of pixels are represented by more than one mac, e.g., the small curved edge feature represented by M_0^6 is also represented by M_6^6 and M_2^6 . Amongst other things, this type of redundancy should provide robustness to faults: e.g., if either of M_0^6 or M_6^6 fails, the other could still represent the presence of the feature.
2. The size and complexity of represented features varies significantly across macs: compare what's represented by M_8^6 with what's represented by M_0^6 .
3. Some active macs do not appear to represent any subset of pixels: this is because such macs, e.g., M_{12}^6 , are only active due to *persistence* (what they actually represent is more complicated and we leave that discussion for later).
4. These features are clearly rather arbitrary looking and do not correspond to the nice, clean, "regular" features that would typically be designed. In fact, these features are the particular features that occurred in the relevant macs' RFs during learning and were assigned to codes stored in the mac.
5. We emphasize that in each of panels, the active feature shown is just one of the features stored in the respective mac's (purple) basis. In general, there is substantial redundancy across the features comprising the basis set of any single mac. This corresponds to what has been termed *overcompleteness*. In future work, we will produce examples/figures like this one, but which also show the complete set of features stored in each mac. This will make it easier to see/analyze the relation to across-mac and within-mac redundancy.

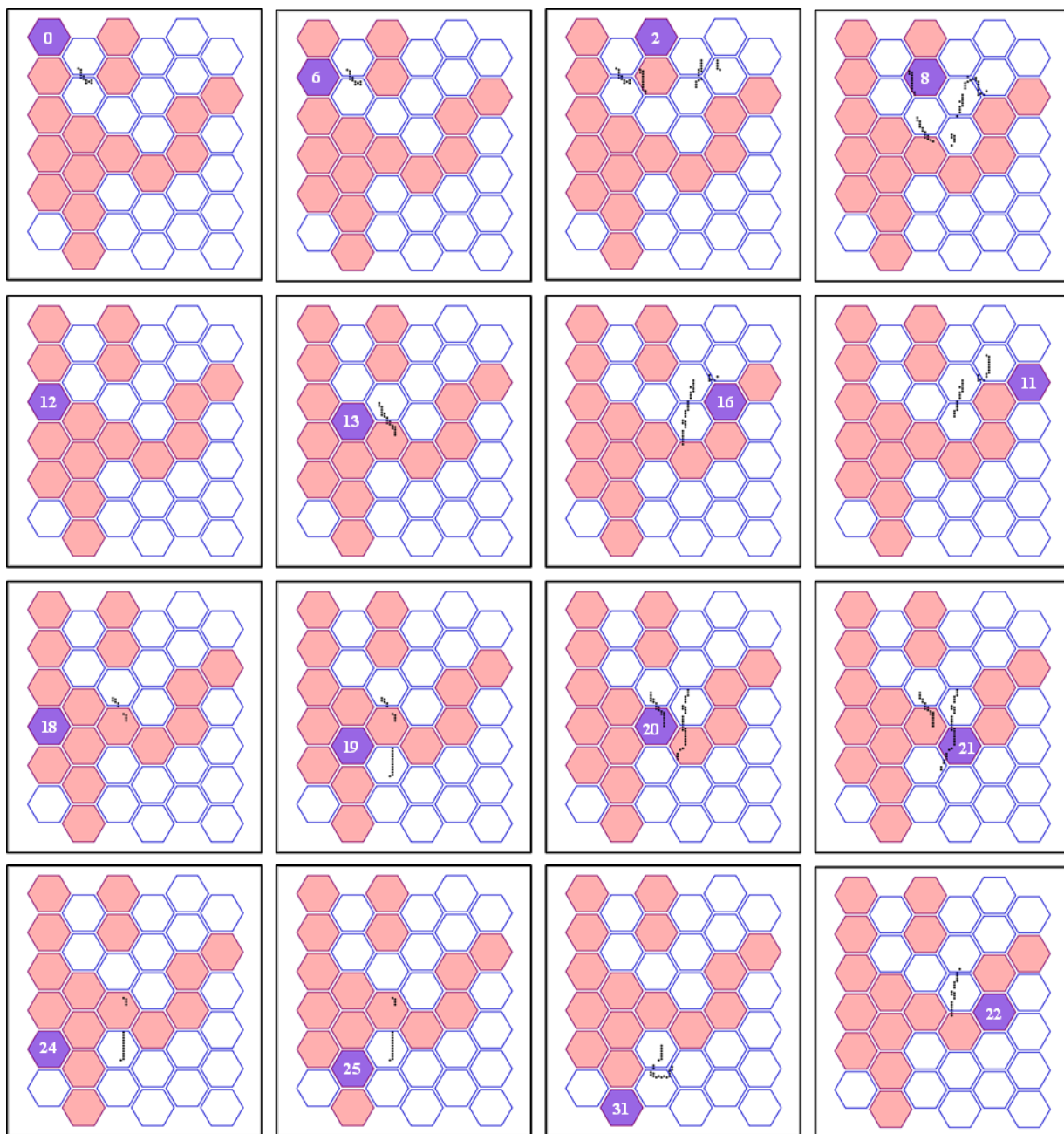


Figure 98: The Portions of the Input, i.e., *Features* or *Parts*, Coded by Each of the 16 Active L6 Macs on Frame 7 of the Input Snippet

APPENDIX E - Relation of Input Accuracy Measure to Recognition Accuracy

Input Accuracy (IA) is defined as the fraction overlap (expressed as a percentage) of a mac's inputs across training and recognition. IA is computed for each active mac at each level and on each frame. IA is computed differently for Level 1 than for subsequent Levels.

IA for Level 1 (L1)

Level 1 macs receive inputs from L0 in the form of a pixel array. Percent overlap is thus computed as the normalized Hamming distance:

$$IA_m = \frac{1}{A(F_U(m))} \sum_{i \in A(F_U(m))} \delta_{train}^{rec}(i) \quad (D-1)$$

where δ_{train}^{rec} is set to 1 if pixel input i is the same across the training and recognition frame and 0 otherwise. $A(F_U(m))$ is the set of active input pixels in mac m 's aperture.

IA for Level 2 (L2) and Higher

Computing IA_m^l for Levels $l = 2, \dots, L$, follows the same approach except that macs that were inactive during *both* training and recognition are discounted. Let $A(F_U(m))$ be the set of macs in the U-RF of mac m of level l , where each input mac was active during either training or recognition or both, for the particular frame considered. The IA is then computed as:

$$IA_m^l = \frac{1}{A(F_U(m)) * Q} \sum_{i \in A(F_U(m))} \sum_{c \in CM} \delta_{train}^{rec}(c) \quad (D-2)$$

where c is the index of the winner in each of the mac's Q CMs. The function $\delta_{train}^{rec}(c)$ here indicates whether the index of the winner in CM c was the same during training and recognition.

E.1 Example Computation for a Level 2 Mac

Whereas the IA is easy to understand for Level 1, the discounting of inactive macs for the IA computed at Level 2 and up is a bit confusing at first. The following example illustrates the rationale for this modification to the usual similarity metric.

Let a mac at L2 have a U-RF consisting of three L1 macs. Let's assume that two of the input macs were silent during both training and recognition. If the third input mac's activity during recognition perfectly matches its activity during training, then Eq. D-2 would indicate a IA of 1. On the other hand, if the training and recognition activity patterns for that mac are completely different, the IA would be evaluated at 0. If inactive input macs were not discounted in the calculations of Eq. D-2, the resulting IA would instead be within the interval $\left[\frac{2}{3}, 1\right]$ depending

on the amount of overlap in the active mac. In other words, by discounting inactive macs the IA coefficient spans the whole range of possible overlap.

E.2 Sample Input Accuracy from a Simulation

A two-level network was run on 64x64 snippets from the preprocessed VIHASI dataset. Level 1 contains 1600 macs, and Level 2, 784 macs. The Level 2 U-RFs contain between 6 and 9 input macs, and each Level 2 mac is active only if between 2 and 4 of its input macs are active. The network is trained and tested on different snippets.

Figure 99 shows a sample distribution of Level 2 mac recognition accuracy percentage (RA%) during recognition as a function of their corresponding IA. As expected, the scatter plot shows a weak tendency for the RA% to be positively correlated with the IA.

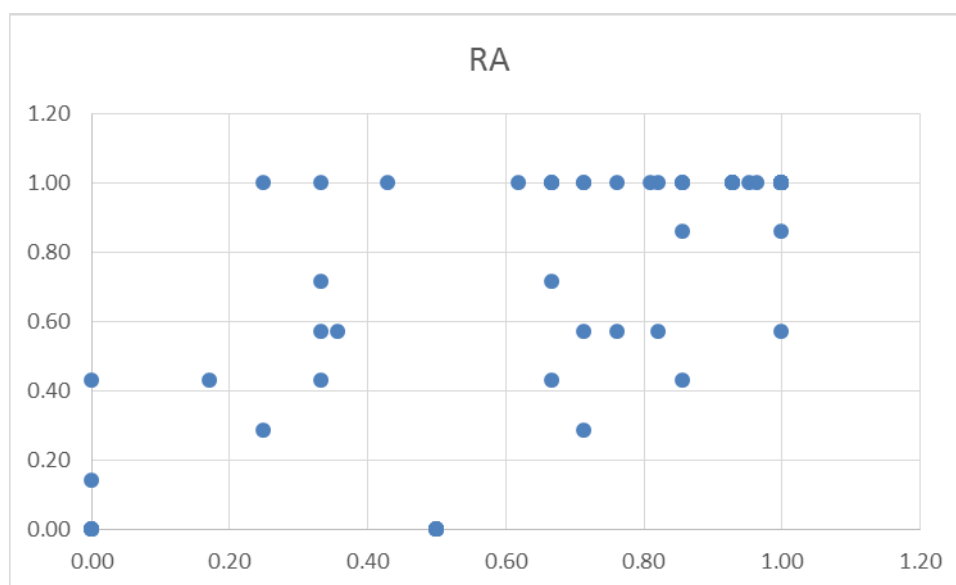


Figure 99: Distribution of RA% as a Function of IA in Level 2 Macs

E.3 Distribution of RA% in a Feedforward Network

In addition to the mean IA computed above, it is useful to look at the distribution of RA% across input macs. This is useful to determine whether a single input mac with high RA can lead an output mac to a high RA% despite the presence of other, low RA input macs. This would be analogous to a Max operator where the state of the output mac is determined by the state of the maximally accurate input mac. The same network as above was used.

Table 25 shows the top eight IA distributions ranked as a function of average RA%. The table can be interpreted as follows. The highest RA% observed tend to occur when all four input macs have an IA between 75 and 100%. Table 25 may show some form of irregularity in the distribution, for example, comparing the third and seventh rows suggests that higher RA% is accomplished by having a single input mac in the topmost bin, followed by two macs in the third bin, and one mac in the second bin, then by having three macs in the topmost bin and one in the third bin. It is not clear at the moment how such a result could arise.

Perhaps more interesting is the comparison with Table 26 which shows the lowest 13 distributions, again ranked by RA%. Note that 13 distributions are shown here because they all had the same average RA%. It is clear that most of the *weight* in the distributions is shifted to the left in Table 26. This is not surprising: in general the more the inputs to a mac vary across training and recognition periods, the lower the RA%.

Table 25. Top 8 IA Distributions Ranked by Average RA%

0 – 25	25 – 50	50 – 75	75 – 100	Average RA%
0	0	0	4	0.95
0	0	0	3	0.90
0	1	2	1	0.85
0	0	0	2	0.81
0	1	0	3	0.76
0	0	1	2	0.73
0	0	1	3	0.72
0	2	2	0	0.71

Table 26. Bottom 13 IA Distributions Ranked by Average RA%

0 – 25	25 – 50	50 – 75	75 – 100	Average RA%
3	0	2	1	0
3	0	0	3	0
5	0	1	1	0
3	2	0	0	0
1	0	2	2	0
1	0	0	4	0
4	0	2	0	0
2	1	1	2	0
3	0	3	0	0
3	0	1	2	0
1	1	2	1	0
1	1	0	3	0
2	2	1	0	0
1	0	1	3	0

However, it can be seen that a substantial number of the lowest distributions have at least one highly accurate input mac. In other words, the presence of an accurate input mac does not seem sufficient to lead to a high output RA%, suggesting that the feedforward network considered here does not behave in a *max*-like way.

APPENDIX F - Spatiotemporal Compositional Hierarchies in Sparsey

The bottom row of Figure 100 shows an 8-frame snippet depicting an “Extend Arm” event. Within individual frames, we can see several components, or parts, e.g., a head, an arm, a hand. For each individual part, we can see sub-parts, e.g., back-of-hand, (curled) fingers, top-edge-of-arm, bottom-edge-of-arm, top-of-head, forehead, cheek. Furthermore, if this is in fact the arm of a person being extended, then we know that the full extent of the arm in some of these frames, must include segments corresponding to the upper arm, to the elbow, to the forearm, and to the wrist, perhaps to the shoulder as well. Figure 100 also shows the mac arrays and the active macs (rose shaded) at each higher level and over all eight frames. In each panel, we also overlay the subset of pixels represented *by the set* of active macs at that level and frame. Not all active pixels end up being represented at L1 by dint of the interactions of the specific L1 mac’s U-RFs and π -bounds. However, because the U-RFs overlap, most pixels end up represented by one or more active L1 macs. The same principles apply all the way up the hierarchy. One place in particular, where a significant portion of the pixels “fall out of the representation” is at L3 on frame 7, i.e., most of the pixels corresponding to the top edge of the arm. However, as the reader can see, those pixels do remain represented at all levels on the surrounding frames. In general, such momentary dropping out of features from a spatiotemporally evolving global representation may be quite tolerable, but this must be investigated thoroughly going forward.

Also, one can readily see delayed activation proceeding up the hierarchy. Finally, note that macs at higher levels persist for progressively longer periods, L2 and L3 codes persist for two frames, and L4-L6 codes persist for four frames (or till end of snippet).

How are the different parts, sub-parts, sub-sub-parts, etc., represented in a multilevel Sparsey network? We will demonstrate how over the next series of figures. At the outset, we remind the reader that Sparsey learns the part representations from scratch and that parts represented at level J generally contribute to more than one “whole” represented at level J+1. We will begin by explaining / demonstrating this in purely spatial patterns and then extend to the spatiotemporal case, i.e., spatiotemporal events, consist of multiple smaller-scale spatiotemporal events, which Sparsey also learns from scratch, and also such that smaller scale events generally participate in multiple higher-level events.

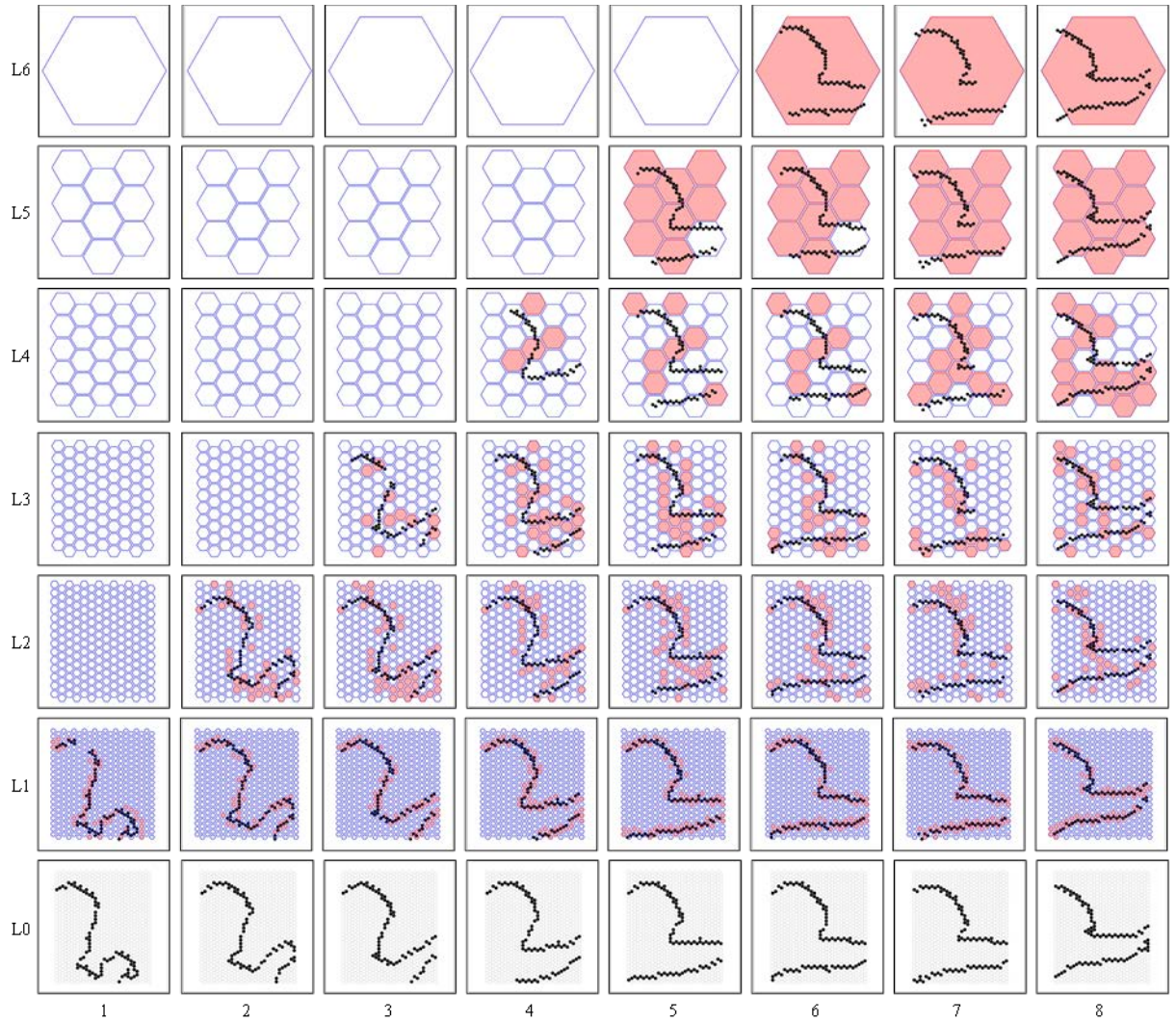


Figure 100: Sets of Active Macs Cross Levels and Frames While Processing an 8-Frame Snippet of an “Extend Arm” Event, Showing Dropping Out of Features At Higher Levels

While Figure 100 shows the subset of pixels collectively represented by the *union* of all active macs at each level and frame, we would also like to know which pixels are represented by any particular level J mac on any particular frame. That is, we want to be able to say that at any given level and frame, this or that mac is representing this or that specific feature, or “part”, of the whole. Figure 101 (center) shows the sets of pixels (black) that are active in the U-RFs of 10 L1 macs highlighted in purple. [The full U-RFs (pixel fields) for each purple mac are shown, in cyan, in Figure 102: each one consists of about 15 pixels.] Each of these 10 small groups of 4-5 active pixels constitutes an L1 feature (approximating a short oriented edge). These 10 features do not overlap only because we chose the purple L1 macs to be sufficiently far apart. However, the full L1 representation of the input pattern is actually the union of all 76 active (rose or purple-shaded) macs. And, in general, any small local portion of the input is “seen by” (is in the U-RF of) several overlying L1 macs. This is illustrated by the four outer panels of Figure 101, which show the active pixels falling within the U-RFs of four nearby L1 macs, M_{69}^1 , M_{90}^1 , M_{108}^1 , and M_{110}^1 , indicated by the red lines. The pixels are numbered to show the overlap. For

example, pixels 3, 4, and 5 are represented by L1 macs 69 and 108, pixel 5 is represented by all four, pixels 8 and 9, only by mac 110, etc. Thus, the overlapped U-RFs entails a substantial degree of redundancy. However, this is a complex form of redundancy in that the multiple local patterns (contexts) in which any given pixel (feature) occurs are generally unique. For example, the SDR code active in mac M_{69}^1 (note that the codes are not shown here) represents pixels 3, 4, and 5, in the *context* of pixels 1 and 2, whereas the code in M_{108}^1 represents pixels 3, 4, and 5, in the *context* of pixel 7. As we will see, this same general scheme of overlapping U-RFs and its concomitant redundancy applies all the way up the hierarchy.

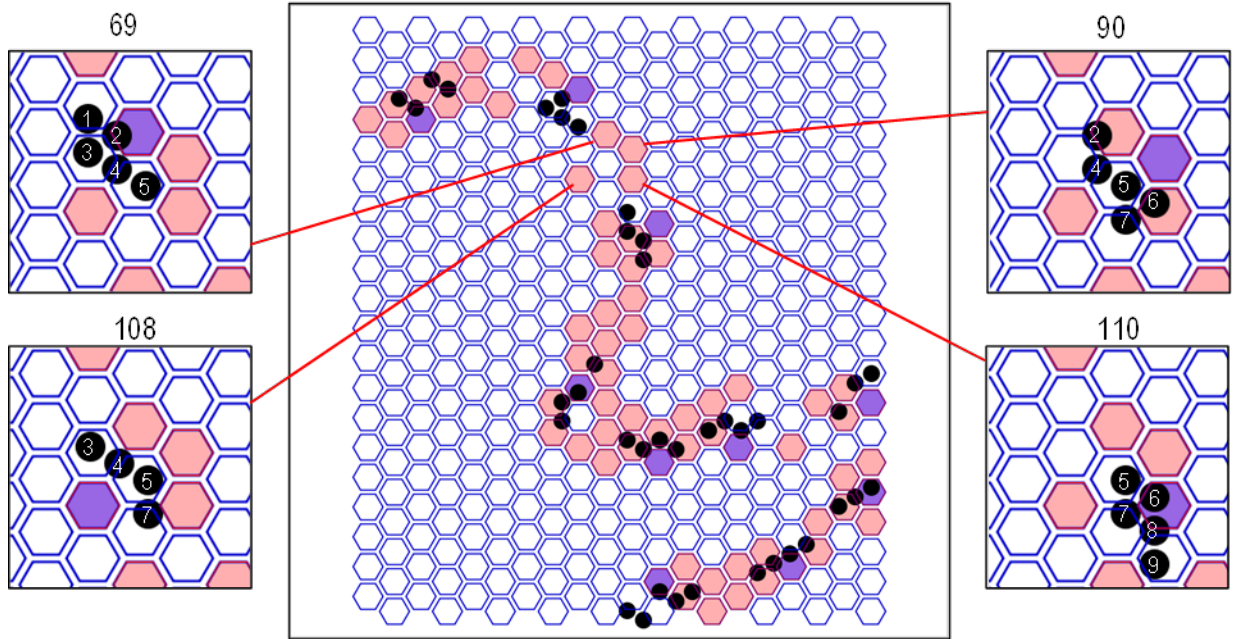


Figure 101: Illustration of Redundant Representation of Features (Parts)

Figure 102 shows a 3D version of the middle pattern of Figure 101, showing only levels L0 and L1 of the 7-level model. In this case, the clumps of cyan pixels show the U-RFs of the aligned overlying purple L1 macs and the blue lines show suggestive subsets of the U-wts from the pixels comprising those U-RFs to the cells of the L1 macs (the cells are not visible). In some cases, the U-RFs of the purple macs abut each other, but none of them overlap. In contrast, in the inset, we show how the U-RFs of the four nearby macs mentioned in Figure 102 (yellow ellipse) do overlap significantly.

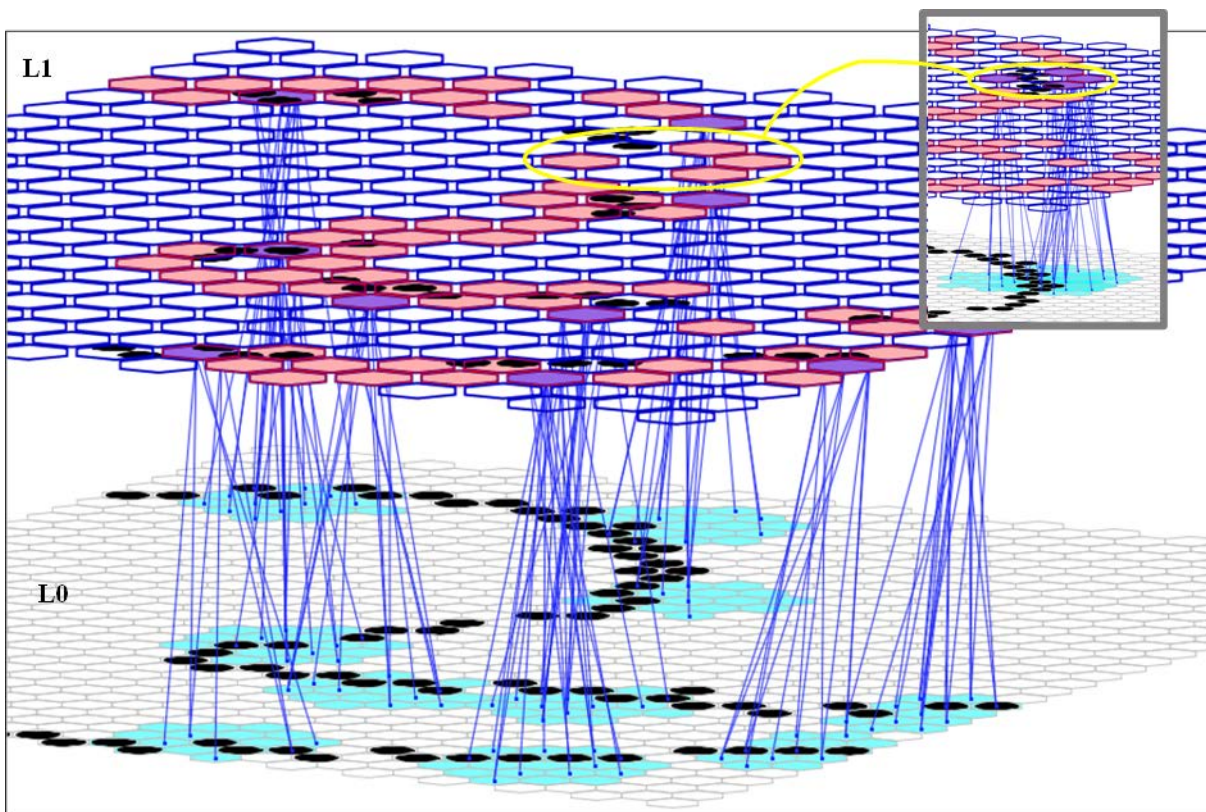


Figure 102: 3D Version of Figure 101's Center Panel to Clarify the Exposition

Figure 103 extends the concepts of Figure 101 and Figure 102 from levels L0 and L1 to L1 and L2. This begins to let us explain how the “parts”, or “features”, become progressively larger and more complex with level, as do the “wholes” composed of those features. The L2 mac grid showing all active macs and all pixels represented at L2 (on frame 4 of snippet) is shown at center of Figure 103b. The L1 mac grid is shown at bottom center for reference. Around the periphery are panels showing the specific sets of pixels represented by seven L2 macs (the purple mac in each panel). Red circles show approximate extent of L2 mac U-RFs at level L0. The reader can see that the features represented by these seven active L2 macs are generally larger and more complex than the 10 L1 features in the center panel of Figure 101. We introduce the convention that a feature name consists of:

A tag naming the principal shape present, e.g., L (straight line), SWC (a southwest corner).

- Table 27 gives a possible list of L2-scale features.
- An optional size measure of shape, e.g., angle in degrees for angle feature, in parens.
- A “.”, followed by the principal approximate orientation, in degrees, of the shape.
- A “.”, followed by an indication of which general sector(s) of the U-RF the centroid of the shape falls within, e.g., central (C), N, NE, E, etc., as in Figure 103.
- The reader can also see the significant overlap between these L2 features, i.e., the 45° line segment represented by the purple mac in the bottom right panel is the same as the lower edge represented by the purple mac in the panel directly above it. However, the

overall features represented by these two macs are clearly different: the bottom right one represents that 45° line *in isolation* while the one above it represents that line *in the context of a nearby 0° edge above it*. As another example, the features in the panels at left and lower left have high overlap. But, while the two overall contexts are different, they are so similar as to be considered representing essentially the same semantic feature.

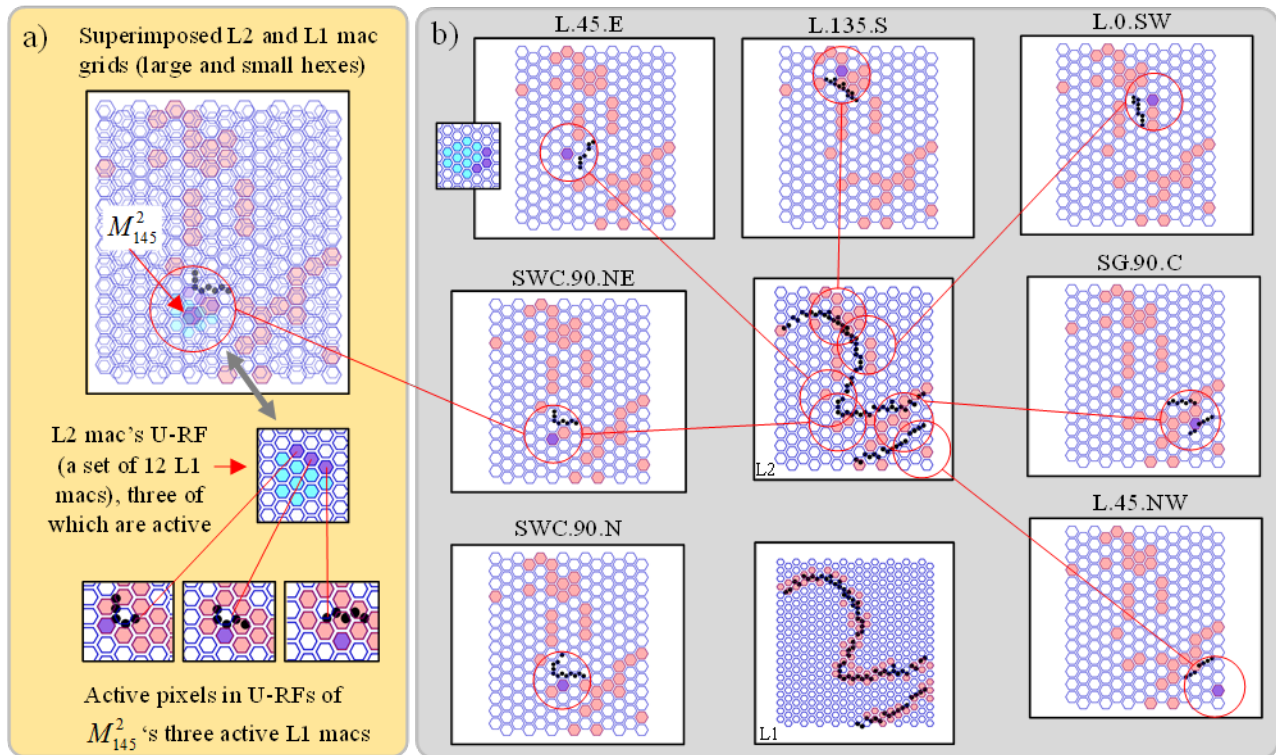


Figure 103: L2 Features are Larger and More Complex than L1 Features

Table 27. List of L2-Scale Simple Shape Features

Feature	Tag	Name	Feature	Tag	Name
	L	Straight Line Segment		Z	Zig-Zag
	SWC, NEC, etc.	Southwest Corner, Northwest Corner, etc.		RL	Radiating Lines
	T	T Junction		CR	Cross
	PG	Parallel Gap		SG	Skewed Gap

We point out that our feature naming convention is deliberately designed to be as *un-semantic* as possible. That is, the reader knows that the 2-segment feature depicted at the right of Figure 103b actually represents the top and bottom edges of an arm, but that's only because the reader brings a huge amount of prior knowledge to interpreting this pattern. However, a "blank slate" instance of Sparsey has no prior knowledge, and a key part of future work is to explain how such

knowledge is acquired over time and examples, most of which are unlabeled, but a small fraction of which are labelled. So we try to assign feature names that carry minimal extra information beyond what could actually be inferred given only the information within the red circle.

Figure 104 now extends our explanation to levels L2 and L3. It has the same scheme as Figure 103b with a grid showing all active L3 macs at center (L1 and L2 are shown below it just for reference) and various specific L3 features around the periphery. To increase clarity, the circles are colored. The circles for three of the panels in the dashed square are not shown in the central L3 panel. Again, the reader can readily see the increasing scale and complexity of the features relative to L2. We would need a substantially richer set of symbolic names of basic shapes at this scale, e.g., as in Table 28. For example, we introduce the notion of an “arc”, as distinct from simply a straight edge, at this scale. The increasing degree of overlap between features represented by neighboring macs is emphasized in the dashed box.

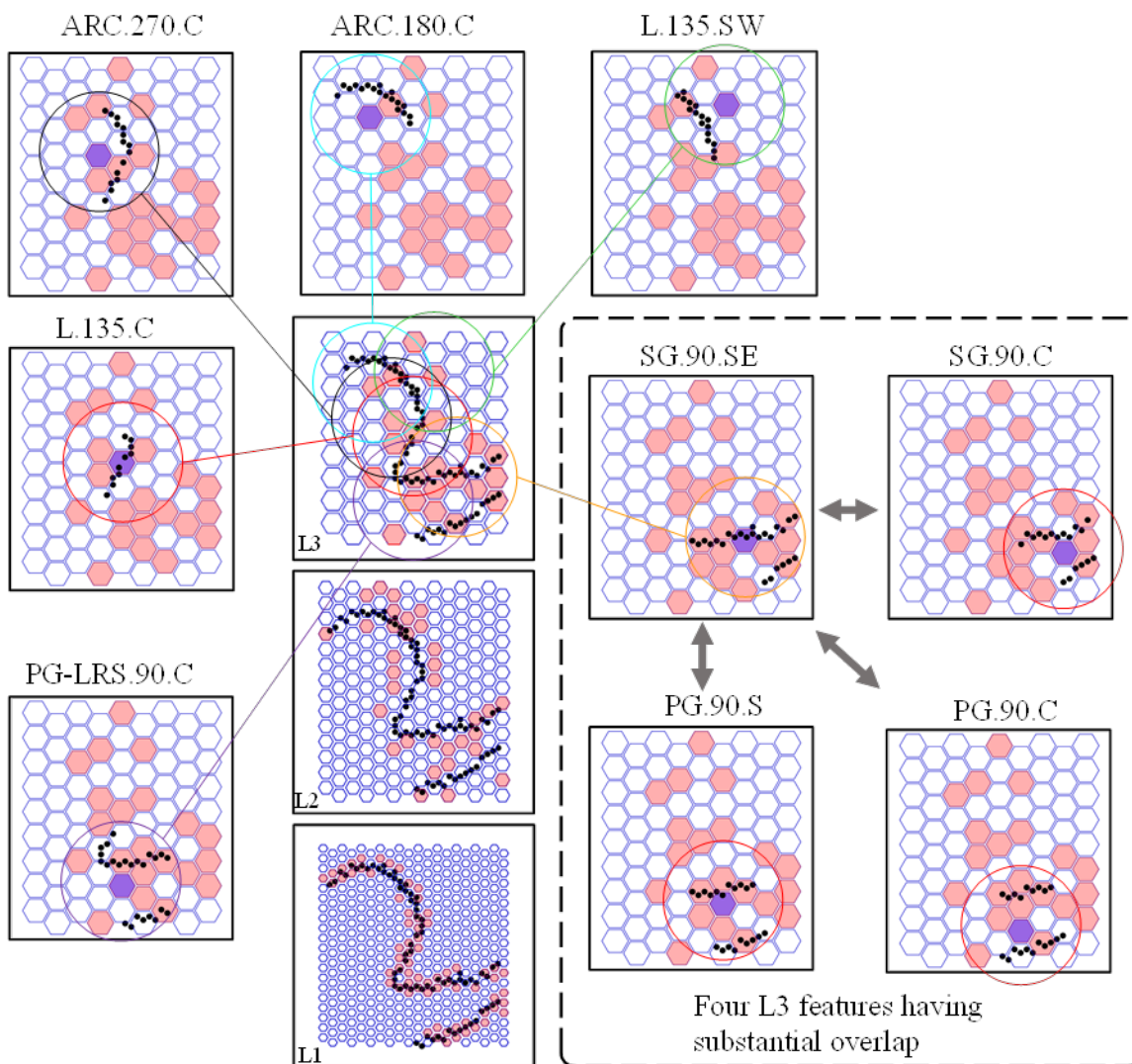


Figure 104: Detail of the Featural Transform Between L2 and L3

Table 28. List of L3-Scale Simple Shape Features








Feature	Tag	Name	Feature	Tag	Name
	L	Straight Line Segment		Z	Zig-Zag
	A	Angle		RL	Radiating Lines
	T	T Junction		CR	Cross
	PG	Parallel Gap		SG	Skew Gap
	ARC	Arc		?	?
	CR	Closed Rounded		CB	Closed Box
	PL3	Three parallel Lines		?	?
	PG-LRS	Parallel Gap with left rear Serif		?	?
	?	?		?	?
	?	?		?	?
	?	?		?	?
	?	?		?	?
	?	?		?	?
	?	?		U	U Channel
	ARC-LRS	Arc with left rear serif		ARC-DLRS	Arc with detached left rear serif

Figure 105 extends the discussion to levels L3 and L4 and again follows the scheme of Figure 103b. Only four of the L4 macs activate on frame 4 of this snippet. The outer panels show the subsets of the L0 pixels represented by each of the four active L4 macs. There is even more overlap (redundancy) of representation at L4 than L3 [despite much fewer macs (and cells) being active]. Furthermore, the size and complexity of the L4 features are much greater than at L3. Note that one of the features (lower left) has two segments, which are highly overlapping with portions of the other three features. One important point to note here is that the bottom contour of the arm drops out at L4. Again, this occurs because the interaction of the U-RF and π bound parameters at L4 fail to allow any L4 macs whose U-RF includes those pixels to activate. But as suggested earlier, this type of drop-out event is usually seen to be transient, with the dropped-out feature typically being actively represented in proximal frames.

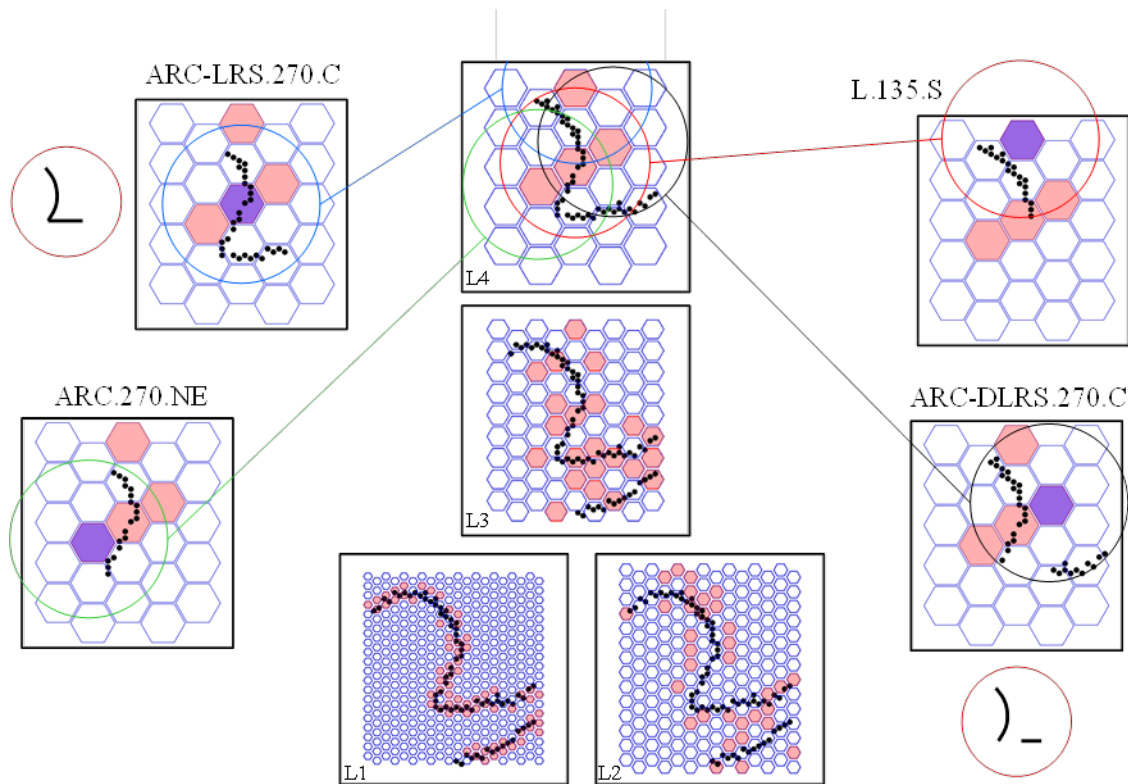


Figure 105: L4 Representation of Frame 4 Consisting of four Large features, Some of which are Fairly Complex

Table 29. List of L4-Scale Simple Shape Features

Feature	Tag	Name	Feature	Tag	Name
	N-FRB	Neck from right of body		ARC-DLRS	Arc with detached left rear serif
	ARC-LRS	Arc with left rear serif			

Figure 106 shows a small subset of instances of progressively larger-scale, more complex, and more overlapped features at successive levels (starting with L2). The L2 mac grid is shown at center with seven copies of the grid, each one showing the U-RF pattern (“feature”) (black pixels) for an L2 mac (highlighted in purple). The idea is that following along paths roughly radiating out from center, we show sequences of features that are the *parts* comprising progressively larger *wholes*. Following the two heavy dashed right arrows, we see two instances of the L3 mac grid, each one showing the U-RF of an L3 mac (purple) whose U-RF includes the L2 mac at the arrow’s origin. Both of these L3 macs fall within the U-RF of the purple L4 mac at far right, whose name is M_{24}^4 . Consequently, the two L3 features are present, as two component “parts” comprising the “whole” feature shown in M_{24}^4 ’s U-RF. Note that M_{24}^4 ’s U-RF includes other active L3 macs as well, which accounts for the other pixels shown in M_{24}^4 ’s U-RF. Consequently, the SDR code active in M_{24}^4 (shown in the inset at top right of Figure 107)

represents the complex feature shown, which could be described in myriad ways, but perhaps most simply as “horizontally extended region with top and bottom edges”.

The reader can continue to follow the heavy dashed arrow to the L5 mac grid at lower right, which shows the U-RF of L5 mac, M_7^5 . This larger-scale feature combines the feature in M_{24}^4 ’s U-RF, with that in M_{16}^4 ’s U-RF, which can be denoted N-FRB.90.NE, i.e., “Neck exiting from right of body, where that entire feature is oriented at 90° and whose centroid is in the NE sector of the U-RF”. We invite the reader to peruse the other features present at the different levels. In general, the idea is that following along dashed paths roughly radiating out from center, we show sequences of features that are the *parts* comprising progressively larger *wholes*. Overall, Figure APPENDIX F - -8 shows the generally increasing size, complexity, and redundancy of the represented features, as well as the how the topological relationships amongst the features represented at any given levels are preserved, though gradually less so with each increasing level. Finally, it shows how a feature at one level generally contributes to multiple features at the next higher level, as seen especially in the bottom and bottom left portions of the figure.

The L6 mac grid (which has only one mac) is shown in inset at top left, along with the subset of pixels represented by the code active in the L6 mac. That subset, which includes almost all the active pixels could, for example, be described as consisting of a “head” and an “arm” feature (and with their implicit spatial relationships), but again, such features are highly loaded with semantics. And, much of those semantics can only have been learned by observing how such features change over time. So we reiterate that we are not suggesting that the codes stored in the macs during the (still very limited) learning phases of our simulations represent high-level features (classes) like, “head”, “arm”, etc. Nevertheless, we do suggest that with more extensive learning, and in larger models, these codes will eventually carry such higher-level semantics, and thus provide a basis for intelligent cognition.

There are two key concepts not captured in the preceding figures. First, they don’t capture the temporal aspect of the features. That is, Sparsey’s macs actually learn (natively) spatiotemporal features, which cannot be fully conveyed by static images. For that reason, we will eventually construct a slide fashioned after Figure 106, which has little animations running in each of the small mac grid panels. This will allow the viewer to get a much better sense of how spatiotemporal features at level J are composed of smaller-scale spatiotemporal features at level J-1. In addition to the progressively larger spatial sizes of the U-RFs at higher levels, persistence increases with level, so the little animations will have progressively longer temporal periods at higher levels.

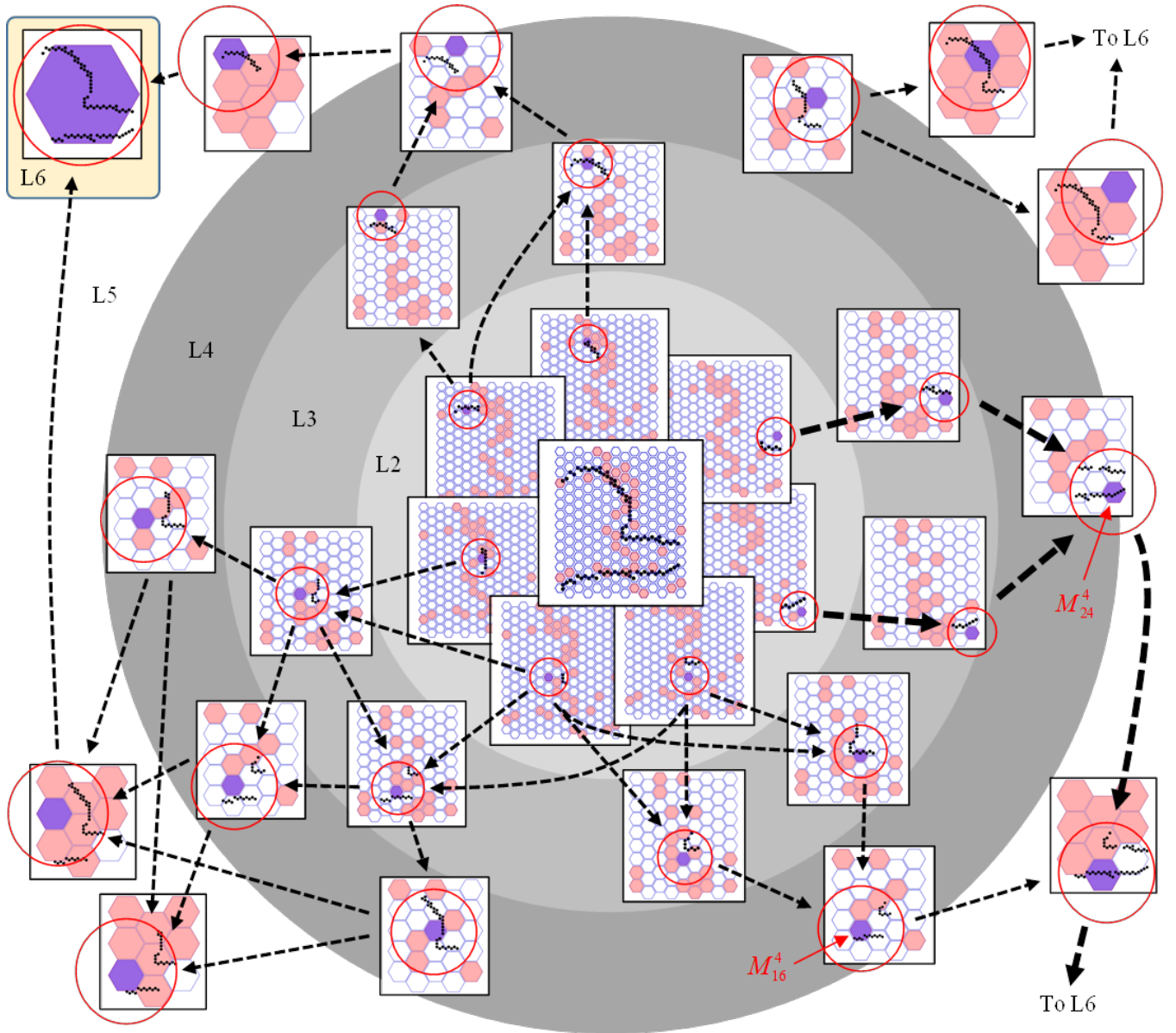


Figure 106: Progressively Larger-Scale, More Complex, and More Overlapped Features at Successive Network Levels (Starting with L2)

The second major concept not portrayed in earlier figures is the relation of the SDR code active in a mac to the actual input in that mac's U-RF. We did show the actual SDR code active in M_{24}^4 at the top right of Figure 107. And, it is the case that that code does represent the current input falling within M_{24}^4 's U-RF. However, we would like to see the spatiotemporal input pattern that occurred in M_{24}^4 's U-RF during the learning phase to which that SDR code was associated. In fact, one can see that there are errors (red cells) in the SDR code shown in the Figure 107 inset. We can truly call these errors because the recognition trial from which the preceding figures were generated involved the exact same input snippet as used during the training trial. However, in general, the model will experience a huge number of future input *moments* that have more or less similarity to learned *moments*, but are not exact duplicates. In

such cases, winners that differ from those that occurred during the closest matching learned moment cannot absolutely be considered errors; that is, such differing winners might correctly reflect the different statistics of the novel current input moment compared to that closest matching learned moment.

Figure 107 shows another visualization to help clarify the U-RF structure across levels of a Sparsey network. The gray “influencing cone” simply shows the expanding U-RF of mac M_{24}^4 at progressively lower levels. At L4, we show the subset of pixels represented by the code (shown in inset at top right) active in L4 mac, M_{24}^4 . As explained on previous occasions, these are the active pixels falling within the region of the input surface (L0) that can influence M_{24}^4 (generally, via multiple vertical sequences of intervening macs). The annotations around the figure explain how the U-RF of a mac at level J, is the union of the macs comprising its immediate subjacent afferent macs at level J-1 (which we can refer to as said mac’s “direct U-RF”), and the same compositional scheme continues down the hierarchy to the input level.

So, in the more general case, where the test snippet \neq training snippet and where nevertheless the code activated is identical or very close to one of the codes activated in the mac for a particular learned moment, we would like to see that learned moment. If the model is to be viable, then it should be the case that in most such instances, the current input moment should be similar to that learned moment. More generally, rather than constraining the prior point such that the code activated during recognition be “identical or very similar” to a code stored during learning, we could simply show the moment corresponding to the learned code having greatest overlap with the code activated during recognition. Or, we could show the moments corresponding to the several learned codes with highest overlap. In any event, adding this capability is essential if we are to be able to view the operating model and judge its ability to retrieve the learned moment that most closely matches the current recognition moment.

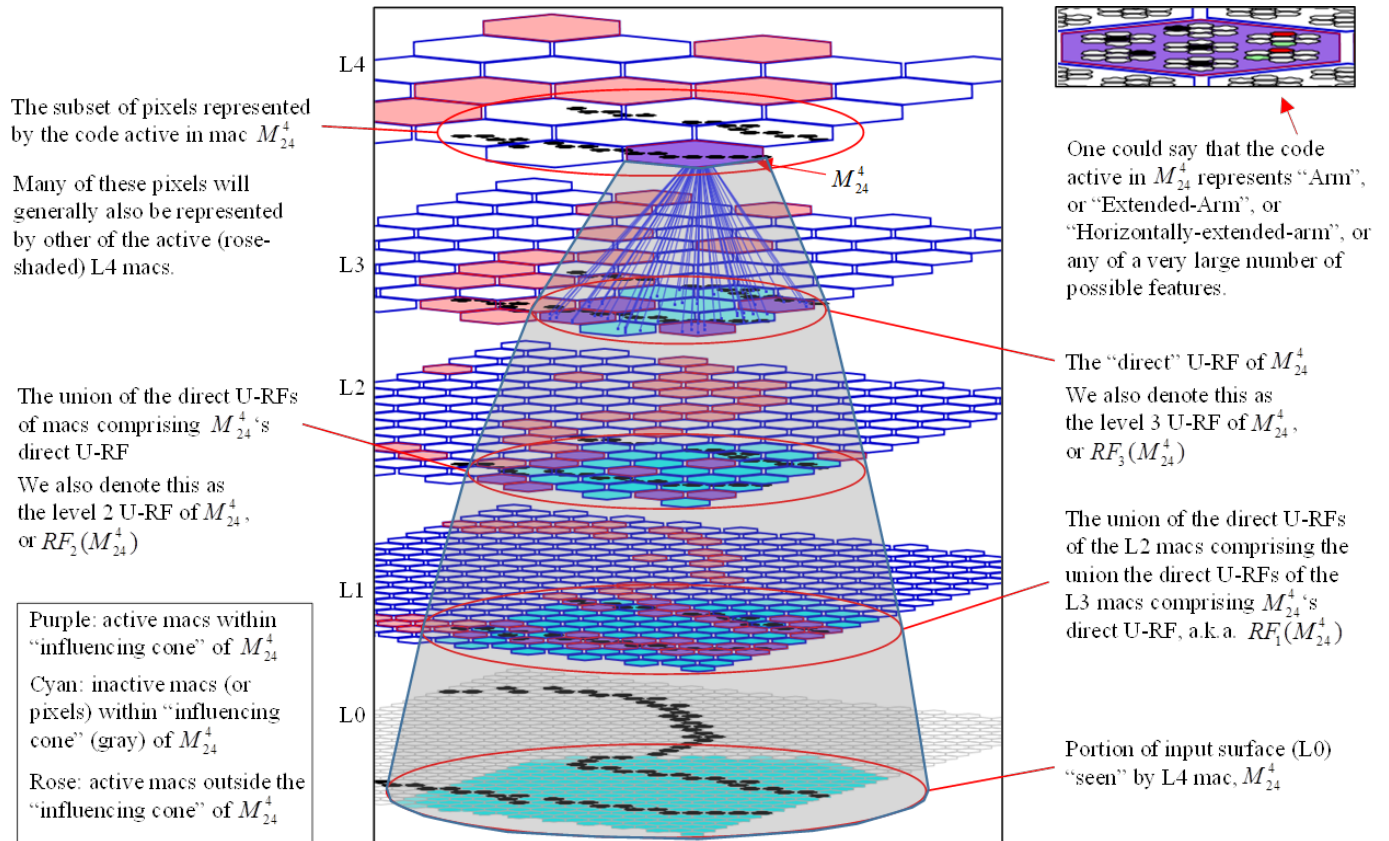


Figure 107: A Visualization Clarifying the U-RF structure Across Network Levels

APPENDIX G - Statement of Work

The project was a two-year seedling effort, which was extended to 2.5 years via an NCTE. The SOW was slightly revised as part of the NCTE. That final-form SOW consists of 21 overall tasks. Table 30 gives our final status on these tasks.

Table 30. SOW Task Final Status

Task	Short description	Status
3.1	Characterize Basic Computational Properties/Capacities of the Core Model	Complete
3.2	Obtain and Preprocess Datasets	Complete
3.3	TEMECOR Benchmarking	Complete
3.4	Transfer Learning	Incomplete
3.5	Parametric Analysis of Specific Contribution of Higher Representational Levels	Incomplete
3.6	Investigate Unsupervised Learning of Low/Mid-Level Visual Features	Complete
3.7	Investigate Cross-Modal Unsupervised Learning	Incomplete
3.8	Investigate Driving Model Based on Only on Global Familiarity (G) Values of Higher Level Macrocolums	Incomplete
3.9	Stored Concepts Simultaneously Ordered on Many Dimensions	Incomplete
3.10	Program Management	Complete
3.11	Characterization/Analysis of Low-Level Visual Features	Complete
3.12	Characterization/Analysis of Mid-Level Visual Features	Complete
3.13	Explore Macrocolumn Weight Freezing Policy	Complete
3.14	Study Effects of Varying Schedule of Persistence Increase Across Levels	Complete
3.15	Investigate Impacts of Modified Learning Policies/Parameters	Incomplete
3.16	Investigate Variation in Network Connection Schemes (Topologies)	Complete
3.17	Develop/Extend Simulation & Testing Infrastructure	Complete
3.18	Investigate Combining Inputs from Multiple Sensory Modalities to Improve Video Event Recognition	Incomplete
3.19	Additional Preprocessing Methods to Provide Cleaner Input to Sparsey	Incomplete
3.20	Compare Performance of Rectangular vs. Hexagonal Grid Topologies	Incomplete
3.21	Conduct Tests on Full-Size Videos, Higher-Level Event Classes, and Segmentation	Complete

G.1 Task List

3.1 Characterize Basic Computational Properties/Capacities of the Core Model: The Contractor shall conduct an initial investigation demonstrating various properties/capacities of learning and recognition performance of TEMECOR SOR model. This task shall involve model instances with a single aperture and single coding modules at each internal level. Key model parameters varied across experiments shall include: input aperture size, number of hierarchical levels, size parameters, control of nonlinear neuron activation function, convergence/divergence of weight matrices, and learning protocol parameters. The Contractor shall sample a tiny fraction of the combinatorially huge space of possible experiments defined by the above parameters. The Contractor shall provide several measures and visualizations in these experiments, including: run-time for training, run-time for retrieving (recognizing), total information (in bits – stored in a module), recall/recognition accuracy, confusion matrices,

precision, recall, and AUC-PR measures.

3.2 Obtain and Preprocess Datasets: The Contractor shall obtain and preprocess the benchmark video datasets so that they can be input into TEMECOR. The Contractor shall develop a flexible standalone preprocessing module to maximally automate production of datasets. Preprocessing research exploration for this task shall include edge-filtering (spatiotemporally local filtering), binarization, the use of intrinsically spatiotemporal features such as optic flow features, known neuronal response profiles of the lateral geniculate nucleus (LGN), the ability to decimate (in time) input videos to any desired output frame rate, the ability to create and convert to any desired spatial resolution, and adding color features to input representation.

3.3 TEMECOR Benchmarking: The Contractor shall conduct a battery of experiments ranging over the parameter space described in 3.2. These experiments shall generally have 300x240 pixel inputs and use models with up to 2000 (50x40) L1 coding modules (each having a 6x6 pixel aperture). The Contractor shall experiment with models with as many as 10 levels, up to several thousand coding modules, several million RUs, and several billion weights. In all cases, the Contractor shall report recognition accuracy using the same measures already present in published results, e.g., Precision-Recall curves. In addition, the Contractor shall report the runtime for training and testing, and other statistics, such as degrees of saturation of weight matrices. The Contractor shall also prepare analyses and visualizations of the RF tuning profiles of modules in a subset of the experiments. The Contractor shall investigate event recognition performance on the benchmarks as a function of input frame rate.

3.4 Transfer learning: The Contractor shall conduct experiments (similar in spirit to those described in (Le, Zou et al. 2011)) in which one trains on one dataset and tests on another. The Contractor shall define a small set of parametric studies, and analyze the transfer obtained as functions of the relevant parameter(s).

3.5 Parametric analysis of specific contribution of higher representational levels: The Contractor shall conduct experiments investigating the advantages of a hierarchical architecture.

3.5.1 To research potential for increased information storage capacity the Contractor shall create a synthetic or naturally-derived dataset of sequences. The Contractor shall compute the amount of information, in bits, contained in each sequence and in the set as a whole: The Contractor shall conduct a parametric search over model space, holding the number of levels J at some low fixed value, e.g., $J=3$ and the total number of weights at W . The Contractor shall follow a training protocol in which, after the presentation of each new sequence, they shall test recall/recognition accuracy on all sequences stored. The Contractor shall iterate this procedure until recall/recognition accuracy falls below a threshold (typically set close to 100%), yielding an estimate, $I(J=3)$ (in bits), of the model's information storage capacity. The Contractor shall conduct another parametric search of model space, but holding $J=4$, and the total number of weights at W . For each model instance tested during this search, the Contractor shall perform the same training protocol as above. The Contractor shall attempt to

show that the $l(J=4) > l(J=3)$. The Contractor shall then repeat this process for $J=5$, etc., with the goal of showing that, all else equal, more levels means higher capacity. The Contractor shall seek to understand general principles/mechanisms underlying this relationship for the particular case of SOR being used with hierarchy.

3.5.2 The Contractor shall try to assess impact of naturalness of Mid-level Features (Synthetic). The Contractor shall create a set of synthetic datasets having specific statistical properties, i.e., spatiotemporal correlations of different scales and orders, or in other word, specific spatiotemporal features. Each such dataset shall be divided into a training set and a testing set. For each dataset, the Contractor shall conduct a parametric search through model space with the goal of finding model instances, or rather spaces of model instances, which can learn the known, multi-scale correlational structure of the dataset as closely as possible. The Contractor shall measure how well the dataset's statistics are learned by measuring recognition accuracy on the test set.

3.6 Investigate unsupervised learning of natural low- and mid-level visual features: The Contractor shall carry out experiments involving the benchmark datasets, which contain numerous instances of natural, high-level events, e.g., hugging, walking, clapping, boxing, golf swinging, horseback riding, skating, etc.

3.6.1 The Contractor shall research visualizations of bases of mid-level coding modules by creating model instances with many levels (~ 10) and training them on natural video from the benchmark datasets. The Contractor shall provide visualizations of the receptive field (RF) tuning profiles for a large sample of coding modules across levels, models, and experiments. A module's RF shall be shown as the exhaustive set (catalog) of the basis elements (features) stored in it. The Contractor shall show the particular SOR code for each such feature as well and shall organize these results by level with the goal of demonstrating the increasing complexity of features with level and the naturalness of features learned at each level.

3.6.2 The Contractor shall compute summary statistics of the features comprising a basis to demonstrate the naturalness of the features learned. For each RU in a module, the Contractor shall compute various summary statistics over the subset of features in whose codes that RU participated. The Contractor shall directly compare these "single-unit" RFs to single-neuron RF profiles from areas throughout the cortical hierarchy.

3.7 Investigate cross-modal unsupervised learning

3.7.1 The Contractor shall extend model simulation for lexical modality. The Contractor shall modify the model simulation to allow an arbitrary number of separate input levels, all of which would be at "LO" level. The Contractor shall implement a preprocessing module that maps text tokens (i.e., of words) into a localist code, i.e., one unit per word. The Contractor shall also develop modules automating extraction of text from video subtitle/caption tracks, beginning with reviewing the specific techniques of (Marszalek, Laptev et al. 2009).

3.7.2 The Contractor shall implement a preprocessing module that uses OTS speech recognition to extract speech (audio tracks) from video and time-align the transcribed text to the video frames.

3.7.3 The Contractor shall extend model simulation for audio modality. To match TEMECOR's current binary input format, the Contractor shall define a binary audio modality representation. The Contractor shall develop an audio preprocessing module which maps digital audio track data into that representation.

3.7.4 The Contractor shall test multi-modal learning by training a model instance on one of the benchmark datasets, presenting the visual and lexical modalities concurrently. The Contractor shall investigate the formation of associations between modalities, which form (as SOR codes) in modules at higher-levels. The Contractor shall compare classification performance with and without the additional modalities and focus on demonstrating the facilitation of learning highly nonlinear categories as described in earlier sections. The Contractor shall carry out a series of studies as described in this subtask, exploring the space of models and using different datasets.

3.7.5 The Contractor shall develop an operational mode of the model which allows cross-modal retrieval testing. Using the models trained in prior subtasks, the Contractor shall present the visual input only of novel examples of given event categories and quantify the model's ability to activate the correct symbol/category representation in the Lexical modality. The Contractor shall also investigate the visual representations that become active given lexical inputs.

3.7.6 The Contractor shall conduct another series of experiments similar to 3.7.3 & 3.7.4 in which the Contractor shall train with three concurrent modalities and test several of the possible retrieval scenarios, e.g., prompt with visual and retrieve lexical, prompt with audio and retrieve lexical, etc.

3.8 Investigate computing global familiarity (G) at higher levels: In the current TEMECOR simulation, each module at every level computes the familiarity, G, of its total input (including top-down, bottom-up, and horizontal inputs).

3.8.1 The Contractor shall implement variable use of G functionality by adding to their simulation, the ability to operate with G being computed only at a specified level and higher, with modules at lower levels using those G values. The Contractor shall implement gradation of the relative use of G values produced at other levels.

3.8.2 The Contractor shall investigate variable use of G functionality by investigating variants of the model in which the G-dependent activation function modulation mechanism operates only at higher levels, and perhaps even with a gradient of efficacy across those higher levels. The Contractor shall repeat several of the synthetic and natural event recognition tasks used in prior tasks while investigating the effect on the tasks of varying the minimum level at which modules compute and use their own G values.

3.9 Stored Features/Concepts simultaneously ordered on many dimensions: In TEMECOR, items (objects, concepts, records) stored in an SOR-based memory/database can be simultaneously physically stored in sorted order on multiple feature dimensions. The Contractor shall conduct a series of simulation-based studies that will demonstrate this property.

3.9.1 The Contractor shall construct a small synthetic dataset of Y sequences (separable) which vary on a predefined set of $M=3$ *separable* (orthogonal) input space features (at least one of which will be spatiotemporal features). The Contractor shall ensure that the features are uncorrelated and/or highly anti-correlated. Thus, the sequences will have *different* orderings on all M dimensions.

3.9.2 The Contractor shall establish simultaneous orderings (separable). The Contractor shall present the Y sequences to the model for single-trial learning. The model used in this experiment shall be of sufficient size (and number of levels) so that only a single code at the topmost level is needed to represent the whole sequence. Assuming persistence doubles with each level above L1, the Contractor shall investigate sequences up to 128 items long with models having eight internal levels. The Contractor shall verify that set of code intersections *simultaneously* respects the M featural orderings.

3.9.3 The Contractor shall construct a synthetic dataset of Y sequences that vary on predefined set of $M=3$ *non-separable* spatiotemporal input space features.

3.9.4 The Contractor shall establish simultaneous ordering (non-separable) by conducting a series of experiments and analyses following the same protocol as in 3.9.2.

3.10 Program Management: The Contractor shall manage both the technical and programmatic aspects of this program including managing technical performance of the team assuring technical progress is made as planned, preparing for program reviews and preparing required program documentation and reports. The Contractor shall notify the government of any program risks in a timely manner.

-----Proposed Year 2-----

3.11 The Contractor shall perform Characterization/analysis of low-level features, corresponding to visual features learned at the first two "cortical" levels, analogous to V1 and V2. The Contractor shall investigate how the model's many parameters affect the features learned, including the numbers of features that can be learned/stored in individual macs and in whole networks. The Contractor shall continue to investigate this parameter space, which includes, #of CMs/mac (Q) for each level, #of cells/CM (K) for each level, parameters specifying the bounds on the numbers of features in a mac's receptive field (RF), which must be active, for the mac to be activated, parameters controlling sensitivity of matching as function of normalized inputs to cell, (i.e., the exponents of U , H , and D in CSA Eq. 3), threshold parameters for backing off to lower-order match computations, whether model uses CSA-based or ML-based recognition, parameters controlling the sigmoid map from a cell's V measure to its final probability of winning, and parameters controlling how the presence of

multiple competing hypotheses in a mac are handled. All of these parameters and many more will affect the (space-time) structures of the features learned, including features in the lowest level macs.

3.12 The Contractor shall perform Characterization/analysis of mid-level features corresponding to visual features learned at the next two "cortical" levels, analogous to V4 to AIT. The same type of investigations can be done for macs at any network level. Due to increasing spatial and temporal size of mac RFs at higher levels, the features being learned shall be of increasing spatiotemporal scale and increasing complexity. The Contractor shall conduct a series of experiments to find systematic relationships between model parameters and the mid-level features that the model learns.

3.13 The Contractor shall explore macrocolumn weight-freezing policy, i.e., "critical periods". There is clear evidence for "critical periods" in biological brains, across multiple modalities. The most natural physical explanation for a critical period is that the set of weights leading to (and perhaps from) a given region of cortex become permanent. We implement the concept of critical period directly in Sparsey via a policy in which, all the weights leading to/from a mac are frozen if the rate of saturation in any one of the projections leading to the mac, e.g., H, U, or D, reaches a threshold. The contractor shall conduct parametric studies of model parameters, and interactions of parameters, that influence the rates of saturation and therefore the schedules/patterns of freezing that obtain. And, there are many functional consequences of any particular schedule/pattern of freezing. We will investigate these principles, parameters, and consequences of weight freezing.

3.14 The Contractor shall study the effects of varying the schedule of persistence increase across levels. Weight freezing requires control of how quickly the various synaptic projections to a mac (e.g., U, H, and D) become saturated. H projections tend to become saturated way sooner than the U- or D-projections. The consequence is that learning is shut off in a mac, when its H-projection is about half saturated but when only a small fraction, -5-10% of the U- or D-weights have been increased. This seems suboptimal from the standpoint that associative memories achieve maximum storage capacity at 50% saturation. The Contractor shall investigate multiple ways of balancing/controlling relative rates of saturation. A primary method is to increase the rate of persistence increase across levels. Basically, if LJ codes stay on for 4x as long as LJ-1 codes that should lead to faster saturation of the U-projection from LJ-1 to LJ, than if LJ codes stay only 2x as long as LJ-1 codes.

3.15 The Contractor shall investigate the impacts of modifying policies/parameters of learning as follows: modify learning law so that strength of weight increase depends on how long the pre- and post-synaptic units have been on and how long the units remain co-active, implement richer synaptic decrease policy, e.g., akin to the decay regime of the STOP learning rule, and investigate properties of parameters controlling size of weight increases as functions of current weight value and permanence values.

3.16 The Contractor shall investigate variation of network connection schemes (topologies). As noted in tech reports submitted under the baseline project, our current simulation environment allows significant control of the extent/pattern of synaptic connectivity

within/between macs at the same and neighboring levels. We have gained some understanding of broad principles, e.g., tradeoffs between degrees of connectivity and rates of saturation, etc. As one example, the greater the increase in persistence in going from network level J-1 to level J, the more slowly the bottom-up (U) projections will saturate, all else equal (because the rate at which new SOR codes are stored in level J macs will be lower). This influences the numbers of basis elements (codes) ultimately stored in all macs involved and indirectly, the ultimate set of (spatiotemporal) classes learned by the model (which will have an associated cost vis-a-vis the classification task). But, all of these considerations will vary as a function of the degree of U- connectivity from level J-1 to J (e.g., if each level J mac has 5 level J-1 macs in its bottom-up (U) RF, vs. if it has only 1, or 9, etc.). The contractor will carry out systematic analyses of interactions of parameters, including connection topologies, and input space statistics. There is a major overarching principle which strongly limits the space of wiring topologies that needs to be searched: specifically, in the brain, neurons typically connect only to ~10⁴ other neurons, i.e., the brain's topology is local. This bodes well regarding eventual mapping of brain-like computation to hardware.

3.17 The Contractor shall continue to develop and extend the simulation & testing infrastructure as follows: Combine the "command line" version of Sparsey and the "GUI" version, add/extend GUI functionality to allow better viewing of learning (increased synapses, wts of synapses, permanences of synapses), etc., viewing of spatiotemporal receptive fields, viewing of spatiotemporal features (i.e., basis elements) stored in macs, viewing of charts showing state CSA variables (u, U, h, H, d, D, V, mu, rho, etc.) for each CM of a mac, automated presentation of graphics for use in papers, and movies for use in presentations/web, automation of unit testing, validation suites, e.g., automatically running a set of benchmark problems, integration of front-end pre-processing (edge-filtering, binarization) into the main application, maintaining a development version and a version that can be made available to users wanting to test Sparsey.

3.18 The Contractor shall investigate combining inputs from multiple sensory modalities to improve video event recognition.

3.18.1 The Contractor shall implement a motion feature representation of video inputs based on histogram of flow (HOF). These motion features constitute a second visual modality of input to the model, the first visual modality being the binary edge features.

3.18.2 The contractor shall produce Sparsey networks that combine (fuse) the two modalities, edges and HOFs, in analogy to the ventral/dorsal (or, "what/where") division of cortical processing. The fusion scheme is depicted in Figure 108. The contractor shall test these multimodal models for classification accuracy on known video benchmark datasets, e.g., KTH, Weizmann, against accuracy achieved by either single modality model.

3.19 The Contractor shall investigate additional preprocessing methods to provide cleaner input to Sparsey.

3.19.1 The contractor shall investigate additional generic preprocessing methods including human actor detection, tighter bounding boxes, and part (limb) labeling to reduce intra-class variation in Sparsey's inputs, in support of better class recognition performance.

3.19.2

The Contractor shall investigate additional preprocessing methods that may yield:

- More complete (unbroken) contours
- Fewer noise pixels, i.e., pixels unconnected with major body outline edges.
- Easier registration of contour segments from one frame to the next.
- Better localization of the single actor in each KTH frame
- Fewer instances of the BB excluding important parts of the image.
- Fewer inclusions of non-informative, e.g. outer frame, edges.

We believe such improvements, though not requiring substantial increase in computational complexity to an overall system [since the algorithms are local (in space and time)], may yield significant increases in Sparsey's ultimate recognition accuracy. We expect improved accuracy because the cleaner data implies reduced intra-class variation without reducing inter-class variation, which makes the classes more separable. Several software packages exist for performing human body localization. The Contractor shall evaluate such packages and if beneficial, will regenerate cleaner binary edge versions of the KTH data, re-test Sparsey with the new data, and analyze performance improvements.

In addition to producing cleaner binary edge imagery, some of these software methods label regions with the specific part, i.e., body limb. In principle, this limb localization information can be used to improve KTH class recognition, e.g., knowing that a particular contour in the upper left quadrant of a frame, which is rotating counter-clockwise, is a "forearm" limb greatly increases the probability that the overall action being performed is "hand-waving". The Contractor shall investigate the feasibility of using such limb localization information to improve recognition accuracy on KTH.

3.20 The Contractor shall compare performance of rectangular vs. hexagonal grid topologies. The Contractor shall compare the two topologies using a large number of problem/network instances and quantify the performance differences. It is expected that hexagonal will significantly outperform rectangular grids in the lower visual levels due to the improved feature size normalization (e.g., in a V1 mac, the number of pixels active in its LGN aperture) afforded by the increased symmetry of a hexagonal vs. rectangular field. However, it is also expected that this advantage will become irrelevant at higher network levels, because such levels are more abstract and less topological in nature. At higher levels, the shapes of macs may in fact become far more irregular and overlapped, which would go a long way towards explaining why macrocolumns have not been reported in frontal brain areas.

3.21 The Contractor shall conduct tests on full-size videos, higher-level event classes, and segmentation. The Contractor shall continue to test on full-size videos as additions/improvements to the model are made in order to validate the model's storage capacity, speed, and the characteristics of higher-level event classes that can and cannot be recognized by the model. The Contractor shall also continue to investigate the model's ability

to segment spatiotemporal objects/events out of the input stream.

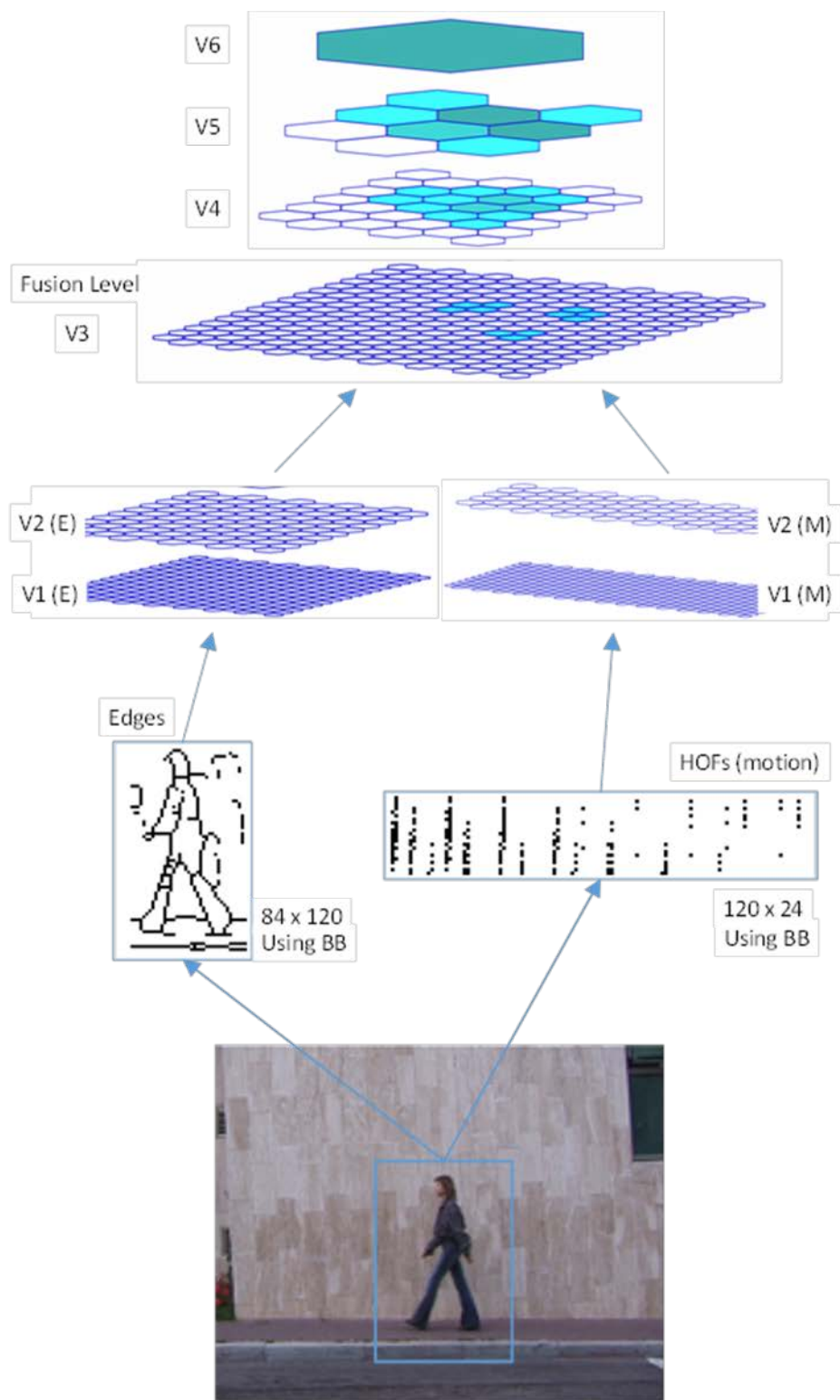


Figure 108: Prospective 2-Modality Fusion Architecture Combining Edge & HOF Features

10. SYMBOL TABLE

Table 31. Major Symbols in CSA Equations

SYMBOL	DEFINITION
$\lambda_{U(t)}$	Power to which U is raised prior to being multiplied with H and D signals. It can vary as a function of time from beginning of the sequence (snippet) being processed.
$\delta(m)$	Persistence, in number of time steps, of mac m . Currently, all macs of a given level have the same persistence.
$\Upsilon(m)$	Age, in number of time steps (frames), of the currently active code in mac m .
U^+	Upper threshold above which a cell's U value is considered 1. H^+, D^+ analogous
π_U^*	Number of active features in a mac's U-RF, which are active in macs with $\zeta \leq B$.
U^-	Lower threshold below which a cell's U value is considered 0. H^-, D^- analogous
π_U	The # of active features in a mac's U-RF.
$V(i)$	Overall local evidence that cell i should become active. Product of functions of $U(i)$, $H(i)$, and $D(i)$.
V_ζ	Threshold for a cell to be considered as part of an active hypothesis
\hat{V}_j	Maximum $V(i)$ in CM, C_j .
G^-	Threshold below which the mac's G value is considered effectively zero.
γ	The sigmoid expansion exponent
$G_{G(t)}$	Average \hat{V} value over a mac's Q CMs. It is a measure of the familiarity of a mac's total input, normalized to [0,1].
χ	The sigmoid expansion factor
η	Range of the V -to- ψ map, which transforms a cell's V value into its relative (within its own CM) probability of winning, ψ .
M_j	Number of macs in Level j .
ζ	The number of maximally active hypotheses, ζ , in a mac.
$a(j, t)$	Activation (0,1) of cell j at time t .
ζ_q	# of cells in CM q with $V(i) > V_\zeta$. Typically, V_ζ is set close to 1, e.g., 0.95.
$F(\zeta(j, t))$	The MCH correction factor $F(\zeta)$ at time t for mac that contains cell j .
B	Threshold on ζ above which we ignore completely signals from the source mac.

SYMBOL	DEFINITION
$F(\zeta)$	The correction factor for increasing the weights of outgoing signals from cells in macs that have <i>multiple competing hypotheses</i> (MCHs), i.e., $\zeta > 1$.
A	Exponent (<1.0) for discounting MCH correction factor when $\zeta > 1$.
$\rho(i)$	The absolute probability of activating cell i in a mac.
$\psi(i)$	The relative probability of activating cell i in a mac.
G_U	G computed based only on the U signals to a mac. Similarly, G_{HUD} is G computed based on all three input vectors, U, H, and D. Similarly, for G_{HU} , G_{UD} , G_{HD} , G_H
$G_{HU}^+(t)$	Threshold below which we back off to the next lower-order (or more generally, the next-considered) version of G . Here, we suggest that this threshold can be a function of time (frame).
$u(i)$, $h(i)$, $d(i)$	Raw sum of weighted signals from cells comprising cell i 's U-RF. $h(i)$, $d(i)$ are analogous
π_U^- , π_U^+	Lower and upper bounds on the number of active features that must be present in a mac's U-RF for that mac to activate.
Q, Q_i	Number of CMs per mac; same but for a specific level, i .
K, K_i	Number of cells per CM; ; same but for a specific level, i
λ_H, λ_D	Analogous to $\lambda_{U(t)}$ except that for now they are not functions of time.
σ_1, σ_2 σ_3, σ_4	Parameters that interact to control overall sigmoid expansivity and shape, e.g., horizontal position of inflection pt., etc.
$U(i), H(i),$ $D(i)$	$U(i)$ is the normalized $u(i)$, to [0,1] range. $H(i), D(i)$ are analogous.
$Active(m)$	Whether mac m is active or not
$M_{2,3}$ M_4^3	The mac at coordinates (2,3) (when the level is unambiguous). Alternate notation: Mac with index "4" at level "3".
U-RF, H-RF, D-RF	U-RF is a bottom-up receptive field. Can be applied to single cells or to whole macs. For cells/macs at L1 the U-RF is a set (or array) of individual binary cells (e.g., pixels). For cells/macs at higher levels, the U-RF is a set (array) of macs. H-RF and D-RF are analogous, but they always consist of a set (array) of macs.

LIST OF ABBREVIATIONS AND ACRONYMS

ACRONYM	DESCRIPTION
ABC	abstraction-based categorization
AFRL	Air Force Research Laboratory
AI	artificial intelligence
CM	competitive module
CONOPS	concept of operations
CSA	code selection algorithm
D	top-down
DARPA	Defense Advanced Research Projects Agency
DCCI	differential correlation of correct and incorrect
DL	Deep Learning
EBC	exemplar-based categorization
GPU	graphics processing unit
H	horizontal
HOF	histogram of optical flow
HOG	histogram of oriented gradient
H-RF	horizontal receptive field
IA	input accuracy
LOO	leave one out
LSH	locality-sensitive hashing
mac	macrocolumn
MBH	motion boundary histogram
MCH	multiple competing hypothesis
MCMC	Markov chain Monte Carlo
MI	machine intelligence
MNIST	Mixed National Institute of Standards and Technology
MSE	mean square error
OP	Overcoding-and-Paring
PAA	post-active activation
PF	projective field
PQA	post-quiescent activation
R	recognition accuracy
RF	receptive field
RU	representational unit
S	symbol
SDC	sparse distributed code
SDR	sparse distributed representation
SISC	similar-input-to-similar-code
SNR	signal-to-noise ratio
SOA	state-of-art
SVM	support vector machine
U	bottom-up
U-RF	bottom-up receptive field
V	vision
WTA	winner-take-all